The Open University of Israel
Department of Mathematics and Computer Science

Clustering of lines

Thesis submitted as partial fulfillment of the requirements towards an M.Sc. degree in Computer Science The Open University of Israel Computer Science Division

by

Author: Tomer Peretz

Prepared under the supervision of Dr. Michael Langberg

Abstract

Let $P = \{p_1, \ldots, p_n\}$ be a set of points in $\mathbb{R}^d$. The k-median problem is the problem of finding k points (facilities) such that the sum of distance from each input point in P to its nearest facility is minimized. The k-median problem is known to be NP-hard. Approximation algorithms for k-median have received a significant amount of research over the past decade, and the current state of the art includes linear time algorithms that return a set of facilities that approximate the optimal solution within a factor of $(1 + \epsilon)$ for any constant ϵ and dimension d.

In this work we study the k-median problem *for lines*. Let $L = \{l_1, \ldots, l_n\}$ be a set of lines. The "k-median for lines" problem is the problem of finding k points (facilities) such that the sum of distance from each line in L to its nearest point is minimized. In this work we study both exact and approximate algorithms for the k-median problem for lines, concentrating on the case in which our input L lies in $\mathbb{R}^2$.

We show that the k-median for lines problem can be efficiently solved for constant values of k, but is NP-complete if k is non-constant. Moreover, we show that solving the k-median for lines problem in $\mathbb{R}^2$ in time less than $poly(n)2^{\epsilon k}$ will imply an algorithm for solving SAT in running time $2^{\epsilon n}$. This implies that it is not likely to improve on the running time of $poly(n)2^{\epsilon k}$ for the k-median for lines problem when ϵ is small. On the positive side, we present an algorithm for finding the 1-median for lines in $\mathbb{R}^2$ in time $O(n^2 \lg n)$, and an algorithm that can find the k-median for lines in $\mathbb{R}^2$ in time $O(n^{2k} \lg n)$.

We then turn to study approximation algorithms for k-median for lines. Using the notion of *bi-criteria* approximation and *core-sets*, we design an $n(c/\epsilon)^{poly(k)}(\log n)^{poly(k)}$ time algorithm that returns a $(1 + \epsilon)$ approximation to the k-median for lines problem on the plane (here, c is a sufficiently large constant).

Acknowledgements

This thesis would not have been possible without the support of many people.

First of all I would like to express my sincere appreciation and gratitude to my supervisor Dr. Michael Langberg for his support and guidance. His invaluable assistance, knowledge and ideas enriched my knowledge and made this thesis possible.

I would also like to thanks Dr. Dan Feldman, for his interesting talks and ideas. A part of this thesis uses the techniques which he was gladly willing to explain.

I would like to thank my parents for teaching me the value of education and for making it possible for me to achieve my educational goals.

Last but not least I would like to thank my family for their support. Especially I would like to my wife Einat for her encourage and for giving me the time to work on this thesis.

Contents

Chapter 1

Intoduction

1.1 General Background

The field of geometric clustering describes grouping sets of geometrical objects while minimizing or maximizing several objective functions. In the standard setting, the input to the problem is a set of points in high dimensional Eucledian space $\mathbb{R}^d$ and a parameter k. The objective is to partition the points into k-groups (or clusters) as to optimize a certain objective function. For example, one function extensively studied during the past decade [HPM04, HPK05, Che06] and most relevant to this thesis, is the k-median objective function. In k-median clustering, for each group a *center* is specified, and the objective is to minimize the sum of distances between the input points and their corresponding centers.

Many variations and restrictions to this standard setting have been addressed. They differ from each other by the type of objects they are clustering, the number of groups, the dimension of the problem and the objective function. Each one of those problems has different motivation and can be related to different fields of research.

For example, we can distinguish between applications based on clustering of geometrical objects in high and low dimensional space. Clustering of geometrical objects in high dimension is more related to fields such as databases [DRSS96], un-supervised learning [WWP00], data mining [Ber06, AGI+92], etc. These problems inherently do not limit the dimension of the geometrical objects. In such cases, to enable feasible algorithms, the solutions running time must have a polynomial dependence on d.

Clustering of geometrical objects in low dimension (for example objects in the plane $\mathbb{R}^2$ or in three dimensional space $\mathbb{R}^3$) is used in problems such as facility location in the plane [CEK02], and the Fermat-Weber problem [BMM03]. These problems assume that the dimension of the input is limited. Studying constant value of the dimension d enables the design of algorithms with running time which has arbitrary dependence, e.g., exponential, in the dimension d at hand. Moreover, such algorithms are strongly based on a variety of special properties that hold in low dimensional Euclidean space.

There has been a significant amount of research in the field of geometric clustering when the data objects to be clustered are points. The most commonly studied clustering objectives are k-center, k-median and k-mean. They are described as follows:

Let $L = \{l_1, ..., l_n\}$ be a set of n points in $\mathbb{R}^d$, and let C be k points $C = \{c_1, ..., c_k\}$. Define $d(l_i, C) = \min_{j=1...k} d(l_i, c_j)$. Here for points x and y, $d(x, y)$ is the Euclidian distance between x and y. If $d(l_i, C) = d(l_i, c_j)$ we say that l_i is associated with cluster j.

- The k-center problem - Given L, find C of size k that minimizes the maximum distance of a point in L to C. Namely find C that minimizes $\max d(l_i, C)$.

- The k-median problem - Given L, find C of size k that minimizes the sum of the distances of a point in L to C. Namely find C that minimizes $\sum d(l_i, C)$.

- The k-mean problem - Given L, find C of size k that minimizes the sum of the square distances of a point in L to C. Namely find C that minimizes $\sum d^2(l_i, C)$.

Since k-center, k-median and k-mean are proven to be NP-Hard problems [MS84] [Pap81], a lot of effort was invested in finding approximation algorithms for these problems. Most of them do not relax the value of k, rather they allow a relaxation in the value of the objective function (if the value of the optimal solution is Opt, then an r-approximation algorithm is guaranteed to return a solution of value at most $rOpt$).

The k-center problem on points has an efficient 2 approximation algorithm. This is close to optimal as it is NP-hard to obtain a 1.82 approximation factor, even in the case the input is given in the plane $\mathbb{R}^2$ [FG88]. For constant d, Euclidian k-median and k-mean can be approximated within a ratio of $(1+\varepsilon)$ in time $O(ndk + f(d, \varepsilon, k))$ [FMS07]. Here f depends polynomially on d and exponentially on $\frac{k}{\varepsilon}$.

1.2 Clustering of lines

The field of line clustering is a relatively new research area. It addresses the task of clustering when the objects to be clustered are lines instead of points. The clustering of lines arises, for example, in the context of handling "incomplete data". It is common to model a data item of d values as a point in $\mathbb{R}^d$. When the data item is missing information on some of its entries, we may consider the data item as a line or hyper plane in $\mathbb{R}^d$ (i.e. the collection of points in $\mathbb{R}^d$ consistent with the partial data item). The clustering of incomplete data is a central problem in statistical analysis, e.g., [All02]. We view the clustering of lines as the most elementary case of incomplete data.

Another example in which the clustering of lines arises is the following (practical) low dimensional problem. A flying probe tester machine is a machine for electrical testing of printed circuit boards. This machine has several probes that touch each electrical line. One may consider the problem of locating the probes in order to minimize the maximum distance between a conductor line and one of the probes or to minimize the total distance that the probes have to move in order to touch all the conductor lines. When looking in small areas of the printed circuit board, the conductor lines can be seen as straight lines. This clearly corresponds to clustering in the context of lines.

The field of line clustering is a relatively new field of research with several open questions that are waiting to be explored. One can study clustering in the context of lines in low or high dimensional space, w.r.t. the k-center, k-means, or k-median objective function, and in the exact or approximate setting.

Previous work on the clustering of lines includes the design of efficient approximation algorithms for the k-center problem on lines when $k \leqslant 3$ [GLS06, GLS10]. Similar to the k-center problem on points, in the k-center problem for lines, one seeks to find k center points that minimize the maximum distance of a line from its closest center. The 1-mean and 1-median clustering of lines was addressed briefly in the work of [HP06]. For k larger than 1, the clustering of both these objective functions remains unstudied and the focus of this thesis addresses the study of the latter for general k (in the plane).

1.3 This Work

In this work we study the k-median for lines problem in the plane. We concentrate on two aspects of k-median for lines in $\mathbb{R}^2$: finding exact and approximate solutions.

Chapter 2 studies the exact solution of the k-median for lines problem in $\mathbb{R}^2$. We start by studying the case where $k = 1$ and then proceed to general k. We suggest algorithms to find the exact solution for both cases. The main idea of this chapter is to prove that there is an optimal solution where all the facilities lie on intersection points between the input lines.

In Chapter 3 we analyze the computational *hardness* of solving k-median for lines exactly. Starting from the study of Megiddo and Tamir [MT82], we show that the k-median for lines problem is NP hard and that solving the k-median for lines problem in $\mathbb{R}^2$ in time less than $poly(n)2^{\epsilon k}$ will imply an algorithm for solving SAT in running time $2^{\epsilon n}$ (the latter being unlikely for small ϵ).

In Chapter 4 we address approximation in the context of k-median for lines in the plane and present an algorithm that guarantees a $(1+\epsilon)$ approximation in time $O(n(\frac{c}{\epsilon})^{poly(k)} \log^{poly(k)} n)$ for some constant c. Our algorithm builds on previous algorithmic paradigms that have been used in the context of approximate clustering. Specifically, we start by presenting a *bi-criteria* approximate solution to our problem (a solution in which both the number of clusters and the solution value are compromised). Here we use ideas and proof techniques from [FFSS07, GLS10]. We then turn to the construction of a *coreset* for the problem of k-median for lines. Namely, using ideas from [HPK05, FFS06] combined with our bi-criteria approximation we construct a small set of lines which represent the original input with respect to the task of clustering. Finally, solving the k-median problem on our coreset, we conclude a solution for the original set of lines.

Chapter 2

Exact algorithms for k-median for lines in $\mathbb{R}^2$

2.1 Finding 1-median for lines

Let $L = \{l_1, \ldots, l_n\}$ be a set of lines in $\mathbb{R}^2$. Let $Dist(p, l_i)$ be the Euclidean distance between a point p and a line l_i. For every point p, let $\varphi(p) = \varphi(L, p) = \sum_{i=1}^{n} Dist(p, l_i)$. In the 1-median problem one has to find a point p^* such that $\varphi(p^*)$ is minimized, namely $\varphi(p^*) = \min_{p \in \mathbb{R}^2} \varphi(p)$. We denote such p^* as the 1-median point and $\varphi^* = \varphi(p^*)$ as the 1-median value.

In the following lemma we show that the intersection points of lines in L includes an optimal center. For simplicity, let us assume that not all the lines in L are parallel. Otherwise the problem can be solved immediately (we will refer to this case later).

Lemma 2.1.1. *Let Q be the set of all the intersection points between two lines from L. Namely $Q = \{l_i \wedge l_j | l_i, l_j \in L\}$. It holds that $\min_{p_i \in Q} \varphi(p_i) = \varphi(p^*)$.*

Proof. We start by proving the following claim.

Claim 2.1.1. *Let L be a set of lines in $\mathbb{R}^2$, and let $\bar{l}$ be any line in $\mathbb{R}^2$ (not necessarily in L) such that $\bar{l}$ intersects at least one line from L. Let $\varphi(L, \bar{l}) = \min_{p \in \bar{l}} \varphi(L, p)$. Then there exists a point $\bar{p}$ on the intersection of $\bar{l}$ and a line in L such that $\varphi(L, \bar{p}) = \varphi(L, \bar{l})$.*

Proof. Let us denote the intersection point of $\bar{l}$ and a line $l_i \in L$ as σ_i and the angle between the two lines as α_i. Let W_p be the sum of distances between any point on $\bar{l}$ and all lines parallel to $\bar{l}$ in L. Let $\bar{L}$ be the set of lines parallel to $\bar{l}$. Let $n' = n - |\bar{L}|$ and let $\{l_1, \ldots, l_{n'}\} = L \setminus \bar{L}$. Let α and β be the two outermost points on $\bar{l}$ in the intersection of $\bar{l}$ and L. Define $f_{\bar{l}}(x) = \frac{1}{\alpha - \beta}(x\alpha + (1-x)\beta)$. For every $x \in [0, 1]$ let $\varphi(L, f_{\bar{l}}(x)) = \sum_{i=1}^{n} Dist(f_{\bar{l}}(x), l_i)$. For simplicity we will refer to $f_{\bar{l}}(x))$ as $f(x)$. Since $Dist(f(x), l_i) = Dist(f(x), \sigma_i) \sin \alpha_i$ then

$$\varphi(L, f(x)) = W_p + \sum_{i=1}^{n'} Dist(f(x), \sigma_i) \sin \alpha_i \tag{2.1}$$

Now consider the derivative of $\varphi(L, f(x))$ according to x, Namely $\varphi(L, f(x))' = \frac{\partial(\varphi(L, f(x)))}{\partial(x)}$.

$$\begin{aligned}\varphi(L, f(x))' &= W'_p + \sum_{i=1}^{n'} (Dist(f(x), \sigma_i) \sin \alpha_i)' \\ &= 0 + \sum_{i=1}^{n'} Dist(f(x), \sigma_i)' \sin \alpha_i.\end{aligned}$$

Let us look at the interval $[\sigma_j, \sigma_{j+1}]$, $0 \leqslant j < n$. Here we assume that the points σ_i appear in the natural order $\sigma_1, \sigma_2, \ldots, \sigma_n$ on $\bar{l}$. It is not hard to verify that the minimum value of $\varphi(L, f(x))$ for $f(x) \in [\sigma_j, \sigma_{j+1}]$ is obtained at the end points σ_j or σ_{j+1}. This follows from the fact that any two points x_1, x_2 such that $f(x_1), f(x_2) \in [\sigma_j, \sigma_{j+1}]$ have the same derivative $Dist(f(x_1), \sigma_i)' = Dist(f(x_2), \sigma_i)'$ for all $0 \leqslant i \leqslant n$. Thus $\varphi(L, f(x))'$ is constant for $f(x) \in [\sigma_j, \sigma_{j+1}]$ which implies that $\varphi(L, f(x))$ obtains its minimum on either σ_j or σ_{j+1} depending on the sign of $\varphi(L, f(x))'$. Let σ_0 be any point on $\bar{l}$ laying "before" σ_1 and σ_{n+1} be any point on $\bar{l}$ "after" σ_n. It is not hard to verify that in the first segment $[\sigma_0, \sigma_1]$ the sign of $\varphi(L, f(x))'$ is negative and in the last segment $[\sigma_n, \sigma_{n+1}]$ the sign is positive. We conclude that $\varphi(L, \bar{l})$ must be equal to one of the intersection points $\{\sigma_j\}_{j=1}^n$. See Figure 2.1 for schematic presentation of $\varphi(L, f(x))'$.

□

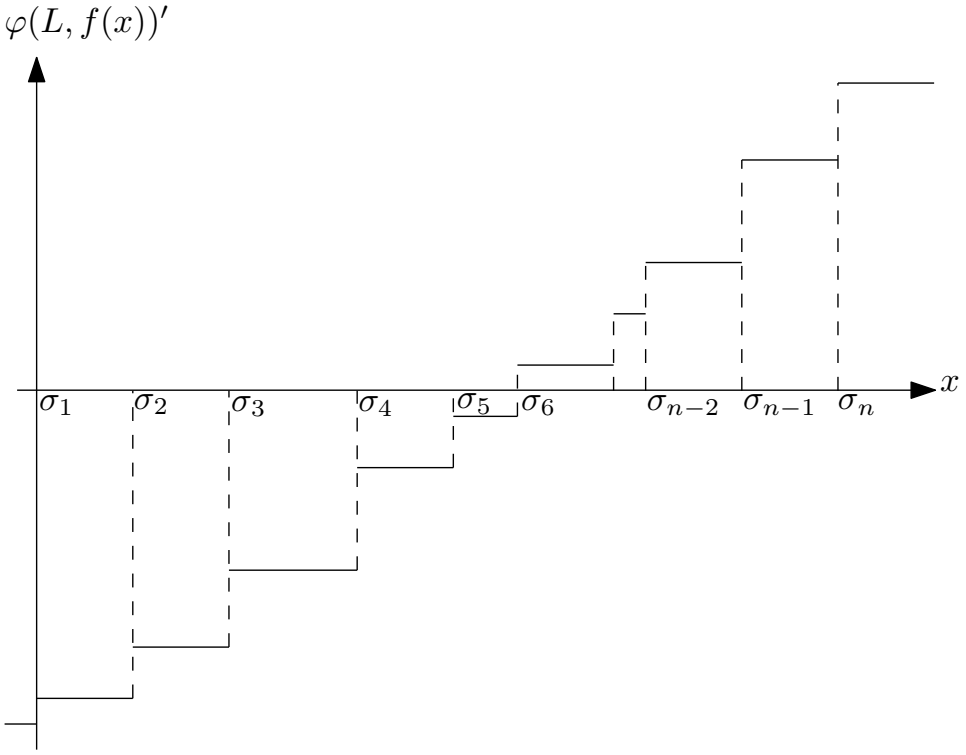

Figure 2.1: The value of $\varphi(L, f(x))'$ changes at each intersection points σ_i by $2 \sin \alpha_i$. Since $Dist(f(x), \sigma_i)' = 1$ for $f(x) > \sigma_i$ and $Dist(f(x), \sigma_i)' = -1$ for $f(x) < \sigma_i$, the difference between $\varphi(L, f(x))'$ for $f(x) > \sigma_i$ and for $f(x) < \sigma_i$ is exactly $2 \sin \alpha_i$.

We now prove Lemma 2.1.1. Let us assume in contradiction that $\min_{p_i \in Q} \varphi(p_i) > \varphi(p^*)$. If p^* is not on any of the lines of L, namely $\min_{p \in L} \varphi(p) > \varphi(p^*)$ then let $\bar{l}$ be any line that passes

trough p^*. For some $j \leqslant n$, let $\bar{L} = \{l_1, \ldots, l_j\}$, be the lines in L that intersects $\bar{l}$. Then by Claim 2.1.1 there is a point $\bar{p}$ on the intersection of $\bar{l}$ and a line from $\bar{L}$ such that $\min_{p \in \bar{l}} \varphi(L, p) = \varphi(L, \bar{p})$. Since p^* is on $\bar{l}$ this contradicts the fact that $\min_{p \in L} \varphi(p) > \varphi(p^*)$.

Otherwise, if p^* is on a line from L, denoting this line by $\bar{l}$ and repeating the argument above, we conclude that there exist a point $\bar{p}$ on the intersection of $\bar{l}$ and another line from L such that $\varphi(p^*) = \min_{p \in \bar{l}} \varphi(L, p) = \varphi(L, \bar{p})$. As $\bar{p} \in Q$, we conclude our assumption.

□

2.1.1 1 median for lines in $\mathbb{R}^2$

Based on the observation of Claim 2.1.1, we can suggest a naive algorithm for the 1-median for lines problem in $\mathbb{R}^2$. The naive algorithm can compute $\varphi(q)$ for every intersection point q between a pair of lines (l_i, l_j) in L. Since there are $O(n^2)$ such points and computing $\varphi(q)$ requires $O(n)$ time, this bring us to a total running time of $O(n^3)$. In this section we improve this running time to $O(n^2 \lg n)$.

Claim 2.1.2. *For a set of lines $L = \{l_1, \ldots, l_n\}$ in $\mathbb{R}^2$, one can find the 1-median of L in time $O(n^2 \lg n)$*

Proof. We start with the case that all the lines are parallel to each other. If this is the case we can reduce the problem to one dimension, projecting each line on a single line perpendicular to all the lines in L. Now each point is represented by a value in $\mathbb{R}$. We have reduce the problem to the 1-median of points in $\mathbb{R}$. This problem can be solved in $O(n \lg n)$ time by sorting all the numbers and finding the middle value (that is the value of order statistics $\frac{n}{2}$). Indeed this value is the 1-median corresponding to the set of points.

If there is at least one pair of lines that is not parallel, it holds that each line in L intersects at least one other line in L.

The main idea of the algorithm is that if for each line, we know the value $\varphi(x)$ for at least one intersection point $x \in Q$, then we can compute the values of all the intersection points of this line in time $O(n \lg n)$. Since each line has at least one intersection point with other lines, we can calculate the values of all the intersection points in time $O(n^2 \lg n)$.

More formally, for each line $l_i \in L$ we sort the intersection points of l_i and other lines in L. Then we partition each line $l_i \in L$ into at most $n - 1$ line segments, defined by the intersection points of l_i and other lines in L according to their appearance on l_i. Let $f_l(x)$, α and β have the same meaning as defined in Claim 2.1.1. For each segment $[\sigma_j, \sigma_{j+1}]$ and a point p on that segement, let $x = Dist(\alpha, p)/Dist(\alpha, \beta)$. Let $g(l_i, p) = \varphi(L, f_{l_i}(x))'$. If we find the value of $\varphi(x_j)$ for a single intersection point, we can compute the value of $\varphi(x_{j+1})$ by $\varphi(x_{j+1}) = \varphi(x_j) + g(l_i, x) Dist(x_j, x_{j+1})$. This means that we can find $\varphi(x_j)$ for all intersection points on l_i in $O(n)$ time. By Lemma 2.1.1, the minimum value from this set is equal to the 1-median in L. Our algorithm is given in Algorithm 1 below. We note that during the execution of the algorithm the value $\varphi(x)$ is computed for all the points $x \in Q$. Our algorithm is described at Algorithm 1.

Running time The loop in line 7 is executed n times. Sorting the intersection points in line 10 of the algorithm, will take time $O(n \lg n)$ for each line, thus $O(n^2 \lg n)$ for all the lines. Finding $\varphi(x_j)$ for all intersection points on a line will take $O(n)$, thus $O(n^2)$ for all the lines. Finding the minimum will take $O(n^2)$. Therefore we can find the 1-median of L in time $O(n^2 \lg n + n^2 + n^2) = O(n^2 \lg n)$. □

Algorithm 1 1 median for $\mathbb{R}^2$

if all the lines are parallel to each other **then**
  return project the lines onto their normal and reduce to 1-median of points in $\mathbb{R}$
end if
$\bar{L} \leftarrow L$
select a point $p \in Q$ and compute $\varphi(p)$.
$\bar{P} \leftarrow p$
while $\bar{L}$ is not empty **do**
  select a line $l \in \bar{L}$ that intersects one of the points p_l in $\bar{P}$.
  $\bar{L} \leftarrow \bar{L} \setminus l$
  compute $\{l \cap l_i | l_i \in L\}$ and order them according to their appearance on l. Denote these points by $P = \{p_1, \ldots, p_s\}$
  for $p_i = p_{l-1}$ to p_1 **do**
    $\varphi(p_i) = \varphi(p_{i+1}) - g(l,p)Dist(p_i, p_{i+1})$.
    $\bar{P} \leftarrow \bar{P} \cup p_i$
  end for
  for $p_i = p_{l+1}$ to p_s **do**
    $\varphi(p_i) = \varphi(p_{i-1}) + g(l,p)Dist(p_i, p_{i-1})$.
    $\bar{P} \leftarrow \bar{P} \cup p_i$
  end for
end while
return the point with the minimal calculated value among the points of $\bar{P}$

2.2 k-median for lines in time $O(n^{2k} \lg n)$

Let $L = \{l_1, \ldots, l_n\}$ be a set of lines in $\mathbb{R}^2$. Let $Dist(p, l_i)$ be the Euclidean distance between a point p and a line l_i. For every set of k points $\Phi = \{\phi_1, \ldots, \phi_k\}$, let $Dist(\Phi, l_i) = \min_{\phi \in \Phi} Dist(\phi, l_i)$ and let $\varphi(\Phi, L) = \sum_{i=1}^{n} Dist(\Phi, l_i)$. In the k-median problem one has to find a set of k points Φ^* such that $\varphi(\Phi^*, L)$ is minimized. We denote such Φ^* as the k-median points and $\varphi^* = \varphi(\Phi^*, L)$ as the k-median value.

Lemma 2.2.1. *Let Q be the set of all the intersection points between two lines from L. Namely $Q = \{l_i \wedge l_j | l_i, l_j \in L\}$. It holds that* $\min_{\phi_1,\ldots,\phi_k \in Q} \varphi(\{\phi_1, \ldots, \phi_k\}, L) = \varphi(\Phi^*, L)$.

Proof. Assume by contradiction that there is a set of k points $\bar{\Phi} = \{\bar{\phi}_1, \ldots, \bar{\phi}_k\}$, such that $\min_{\phi_1,\ldots,\phi_k \in Q} \varphi(\{\phi_1, \ldots, \phi_k\}, L) > \varphi(\bar{\Phi}, L)$, and there is a point $\bar{\phi} \in \bar{\Phi}$, such $\bar{\phi} \notin Q$. Here we assume that $\bar{\phi}$ can not be replaced with one of the points in Q. Let $\bar{L} \subset L$ be the set of lines that $\bar{\phi}$ is the closets point to them among the points in $\bar{\Phi}$, namely $\bar{L} = \{l \in L | Dist(\bar{\Phi}, l) = Dist(\bar{\phi}, l)\}$.

By Lemma 2.1.1 there is a point $p \in Q$ such that $\varphi(\bar{L}, p) \leqslant \varphi(\bar{L}, \bar{\phi})$. This in contradiction that $\bar{\phi}$ can not be replaced with a point from Q. □

In order to find the k-median for lines we can use a naive algorithm that looks at all $\binom{n^2}{k}$ possibilities to take k points out of n^2 possible intersection points (actually there are only $\binom{n}{2}$ intersection points, but we use n^2 to simplify notation). For each possible solution the algorithm calculates the cost of that solution and eventually takes the solution with the minimal value. The running time for such an algorithm is $O(n^{2k+1})$.

We will now present an algorithm that finds the k-median for lines in time $O(n^{2k} \lg n)$. This is an improvement over the trivial algorithm described above.

We start with the case that all the lines are parallel to each other. If this is the case we can reduce the problem to one dimension, projecting each line on a single line perpendicular to all the lines in L. Now each point is represented by a value in $\mathbb{R}$. We have reduced the problem to the k-median of points in $\mathbb{R}$. This problem can be solved in $O(n^3 k)$ time. We will refer to this problem (and suggest an efficient algorithm) in Section A.2 of the Appendix.

Let $\Phi = \phi_1, \ldots, \phi_k$ be the k points of the optimal solution. Assume that we know the first $k-1$ points of the optimal solution $\phi_1, \ldots, \phi_{k-1}$. Moreover since we know by Lemma 2.2.1 that all the points of Φ are on the lines of L, assume that we know the line l that contains the last point of the solution, namely ϕ_k. Now we will show how to find ϕ_k.

Let $l_1, \ldots, l_m \in L$ for $m < n$ be the lines that intersects l and let $q_1, \ldots, q_m$ be the intersection points of those lines and l. Each l_i intersects l, therefore there is an interval $[a_i, b_i]$ on l such that $\forall p \in [a_i, b_i]$ $Dist(l_i, p) \leqslant \min_{j \in 1,\ldots,k-1} Dist(l_i, \phi_j)$. Namely if ϕ_k will be any point in this interval, the line l_i will be closer to ϕ_k then to any other point in $\{\phi_1, \ldots, \phi_{k-1}\}$. See Figure 2.2.

For a point x on l, let $\bar{L}_x$ be all the lines $l \in L$ such that $\varphi(l, x) < \varphi(l, \{\phi_1, \ldots, \phi_{k-1}\})$. Notice that $\varphi(\Phi, L) = \varphi(x, \bar{L}_x) + \varphi(\{\phi_1, \ldots, \phi_{k-1}\}, L \setminus \bar{L}_x)$. We want to find x which minimizes $\varphi(x, \bar{L}_x) + \varphi(\{\phi_1, \ldots, \phi_{k-1}\}, L \setminus \bar{L}_x)$.

Let $price_k(x) = \varphi(x, \bar{L}_x)$ and $price_{other}(x) = \varphi(\{\phi_1, \ldots, \phi_{k-1}\}, L \setminus \bar{L}_x)$. We will now present an algorithm that finds the k-median for lines in time $O(n^{2k} \lg n)$.

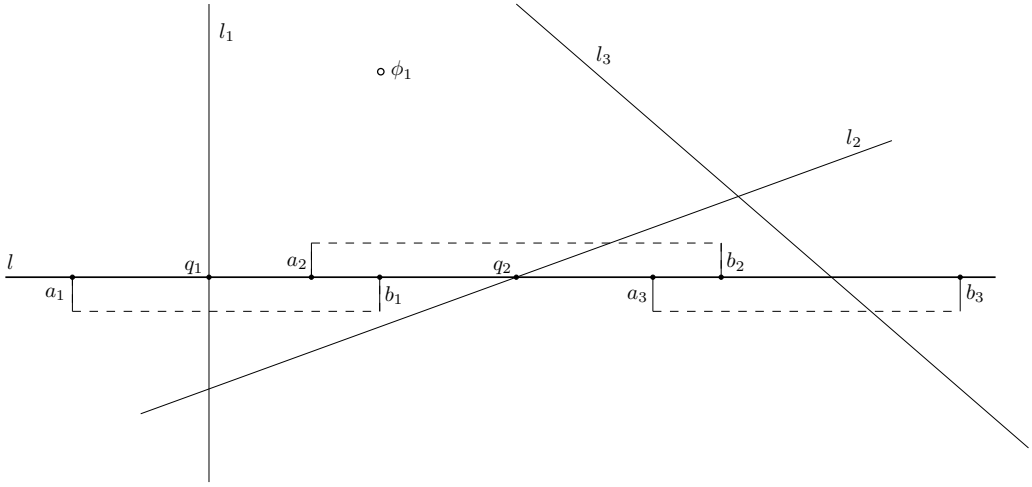

Figure 2.2: An example of lines from L and the corresponding intervals created on l (only a subset of the facilities appear in the figure). In this example, if ϕ_k is between a_2 and b_1 then ϕ_k will be the closest facility for both l_1 and l_2.

2.2.1 The Algorithm

Our algorithm is described in Figure 2.2. In what follows we explain the notation in the algorithm. For each line l_i we call a_i its entrance point, b_i its exit point and q_i its intersection point with respect to l. We define $dist_{outer}(l_i)$ as the minimum distance of l_i to $\{\phi_1, \ldots, \phi_{k-1}\}$, namely $dist_{outer}(l_i) = \varphi(\{\phi_1, \ldots, \phi_{k-1}\}, l_i)$.

Without lost of generalization, we assume an orientation of l. Our data structure will collect all entrance points, exit points and intersection points; and order them along the direction of l.

The algorithm we suggest will travel along l in its direction, maintaining $price_k(x)$ and $price_{other}(x)$. In order to maintain $price_{other}(x)$ we need to remove the value $dist_{outer}(l_i)$ when we meet an

entrance point, and to add it when we meet an exit point. $price_{other}$ is initialized for each $\{\phi_1, \ldots, \phi_{k-1}, l\}$ as the sum of distances of all lines beside l, to the nearest point among $\phi_1, \ldots, \phi_{k-1}$. $price_k(x)$ is initialized with 0. In order to maintain $price_k(x)$ we should keep a function dir_k that holds the derivative of $price_k(x)$. For each l_i let α_i be the angle between l_i and l on q_j. In general $price_k(x) = \sum_{l_i \in \bar{L}_x} Dist(q_i, x) \sin \alpha_i$, thus the derivative of $price_k(x)$ is $\sum_{l_i \in \bar{L}_x} \theta_i \sin \alpha_i$, where $\theta_i \in \{1, -1\}$ depends on whether $x \in [a_i, q_i]$ or $x \in [q_i, b_i]$. As $price_k(x)$ is linear between two consequent points, it needs to be updated each time we meet an entrance point, exit point or intersection point.

Algorithm 2 k-median for lines

1: **for** each $\{\phi_1, \ldots, \phi_{k-1}\}$ in Q **do**
2: $price_{all} \leftarrow \sum_{i=1}^{n} dist_{outer}(l_i)$
3: **for** each line l **do**
4: $price_{other} \leftarrow price_{all} - dist_{outer}(l)$
5: $price_k \leftarrow 0$
6: $dir_k \leftarrow 0$
7: $list \leftarrow$ order entrance, exit and intersection points on l
8: **while** $list$ is not empty **do**
9: $x_j \leftarrow list$ pop next point
10: $l_i \leftarrow$ line corresponding to x_j
11: **if** x_j is an entrance point **then**
12: $price_{other} \leftarrow price_{other} - dist_{outer}(l_i)$
13: $price_k \leftarrow price_k + dir_k Dist(x_j, x_{j-1}) + \varphi(l_i, x_j)$
14: $dir_k \leftarrow dir_k - \sin \alpha_i$
15: **end if**
16: **if** x_j is an exit point **then**
17: $price_{other} \leftarrow price_{other} + dist_{outer}(l_i)$
18: $price_k \leftarrow price_k + dir_k Dist(x_j, x_{j-1}) - \varphi(l_i, x_j)$
19: $dir_k \leftarrow dir_k - \sin \alpha_i$
20: **end if**
21: **if** x_j is an intersection point **then**
22: $price_k \leftarrow price_k + dir_k Dist(x_j, x_{j-1})$
23: $dir_k \leftarrow dir_k + 2 \sin \alpha_i$
24: **if** $price_k + price_{other} < min$ **then**
25: $min \leftarrow price_k + price_{other}$
26: $points \leftarrow \{\phi_1, \ldots, \phi_{k-1}, x_j\}$
27: **end if**
28: **end if**
29: **end while**
30: **end for**
31: **end for**
32: **return** min, points

2.2.2 Running time

The first "for" loop (line 1) is done $O(\binom{n^2}{k-1}) \leqslant O(n^{2k-2})$ times. In line 2 we calculate $dist_{outer}$ for each line. $dist_{outer}$ can be computed in $O(k)$ time. Thus the total cost of line 2 is $O(n^{2k-1}k)$. The

second for loop (line 3) is done $O(n)$ times, therefor each line inside the loop is executed $O(n^{2k-1})$ times. Ordering the points (line 7) is done in time $O(n \lg n)$, thus a total of $O(n^{2k} \lg n)$. The While loop (line 8) is done $O(n)$ times which gives us a total time of $O(n^{2k})$. This bring us to total of $O(n^{2k} \lg n)$.

Chapter 3

k-median for lines: lower bounds

3.1 Hardness of computing k-median for lines

We showed that the k-median for lines problem can be solved in $\mathbb{R}^2$ in time that is polynomial in n, but exponential in k. We will now show that the k-median for lines problem is NP-Hard when k is not fixed in advanced and is part of the input.

Megiddo and Tamir prove [MT82] that:

Theorem 3.1.1 ([MT82]). *Let $l_1, \ldots, l_r$ be a set of straight lines. Finding a set of points $\{(x_1, y_1), \ldots, (x_p, y_p)\}$ of minimum cardinality such that each l_j contains at least one (x_i, y_i), is NP-Hard.*

Lemma 3.1.1. *k-median for lines in the plane in NP-hard.*

Proof. We use a simple reduction from the problem studied in [MT82] to the k-median for lines problem. Let us assume in contradiction that the k-median for lines problem is not NP-Hard. We design an algorithm for the problem discussed in [MT82]. Our algorithm starts with $k = 1$ and checks whether the k-median equals to zero. If not, then increase k by one until the solution for the k-median is zero. The first k for which the k-median is zero is the minimum cardinality of a set of points $\{(x_1, y_1), \ldots, (x_p, y_p)\}$ such that each l_j contains at least one (x_i, y_i). This will solve the problem of [MT82].

This proves that k-median for lines in $\mathbb{R}^2$ is also an NP-Hard problem. Moreover these results imply that approximation within any multiplicative factor is also NP-Hard. Namely, a multiplicative approximation of a solution of value zero is also zero. □

3.2 Lower bounds on the running time of k-median for lines

We have shown that k-median for lines in the plane is NP-hard. We will now present a lower bound on the running time of k-median for lines. For this purpose we will look at a reduction from the 3-SAT problem. The 3-SAT problem is a Boolean satisfiability problem, it is expressed as a boolean AND of clauses. Each clause is expressed as a boolean OR of 3 boolean literals. The 3-SAT problem is an NP-Hard problem, it was well studied and it is believed that the 3-SAT problem can not be solved in time $O(2^{\epsilon n})$ for constant value of ϵ approaching zero, see for example [Woe03]. In what follows $poly(n)$ refers to n^c for some constant c.

Lemma 3.2.1. *Let $\epsilon > 0$, k-median for lines in the plane can not be solved in time $poly(n)2^{\epsilon k}$ unless 3-SAT can be solved in time $2^{\epsilon n}$.*

Proof. In their paper [MT82] Megiddo and Tamir define two problems. The point covering problem (PC) and its dual, the line covering problem (LC). The PC problem is defined as follows: A set of

points $(x_1, y_1), \ldots, (x_p, y_p)$ (x_i, y_i rationals, $i = 1, \ldots, p$) is given. Find a collection of straight lines $\{l_1, \ldots, l_r\}$ of minimum cardinality, such that (x_i, y_i) lies on at least one l_j.

The LC problem is defined as follows: A set of straight lines $l_1, \ldots, l_p$ is given. Find a set of points $\{(x_1, y_1), \ldots, (x_r, y_r)\}$ of minimum cardinality such that each l_j contains at least one (x_i, y_i).

Megiddo and Tamir show a reduction from 3-SAT to PC, in order to prove that PC is NP-Hard. For a given 3-SAT formula φ with m clauses and n variables, the reduction creates in polynomial time, $m^2 n$ points and shows that if φ is feasible, then all the points can be covered by nm lines, otherwise more lines are needed. The reduction constructs for each 3-SAT formula $\varphi = E_1 \wedge E_2 \wedge \cdots \wedge E_m$, with each E_j of the form $x_j \vee y_j \vee z_j$ (here, $x_j,\ y_j,\ z_j \in \{v_1, \ldots, v_n, \bar{v_1}, \ldots, \bar{v_n}\}$), the following instance of PC (See Figure 3.1):

- For each clause E_j we create a point P_j, such that no three points are on a straight line.
- Each pair of variables $(v_i, \bar{v_i})$ is presented by grid of m^2 points p^i_{kl} $(1 \leqslant k, l \leqslant m)$, such that:
  - For each $v_i \in E_j$ only the points $\{p^i_{1j} \ldots p^i_{mj}\}$ and P_j are on a straight line. We denote the points $\{p^i_{1j} \ldots p^i_{mj}\}$ as P_{ij}
  - For each $\bar{v_i} \in E_j$ only the points $\{p^i_{j1} \ldots p^i_{jm}\}$ and P_j are on a straight line. We denote the points $\{p^i_{j1} \ldots p^i_{jm}\}$ as $\bar{P}_{ij}$

In [MT82] it is shown that φ is satisfiable iff the entire collection of points $\{p^i_{kl}\} \cup \{P_1, \ldots, P_m\}$ can be covered by nm lines.

If φ is satisfiable, then the solution suggested consists of the following mn lines. Let γ be an assignment that satisfies φ

- If v_i is true in γ, we create m lines L_{ij} $(0 \leqslant j \leqslant m)$, such that L_{ij} covers $\{p^i_{1j} \ldots p^i_{mj}\}$.
- If v_i is false in γ, we create m lines $\bar{L}_{ij}$ $(0 \leqslant j \leqslant m)$, such that L_{ij} covers $\{p^i_{j1} \ldots p^i_{jm}\}$.

It is easy to verify that this setting has mn lines and covers all the points in $\{p^i_{kl}\} \cup \{P_1, \ldots, P_m\}$. The solution is depicted in Figure 3.1.

If φ is not satisfiable, roughly speaking, the main idea of the proof of [MT82] is that each $m \times m$ array can be covered by m lines iff all the lines are in the same direction (v_i or $\bar{v_i}$) and thus correspond to an assignment to v_i. Namely, in order to cover all the points with nm lines, the points in each $m \times m$ grid should be covered by at most m lines. Since each array represents a variable v_i, if φ is not satisfiable, by the construction not all the points $\{P_1, \ldots, P_m\}$ will be covered.

This reduction can be improved from the setting in which one is given $O(m^2 n)$ points which should be covered with nm lines to one in which $O(m^2)$ points should be covered with $3m$ lines.

Instead of creating m^2 points $\{p^i_{kl}\}$ for each variable v_i, we create a subset Q_i of points defined as follows. Let I be a set of clauses in which v_i or $\bar{v_i}$ appear. Our set Q_i will be equal to $\{p^i_{kl}\}_{E_k, E_l \in I}$. Our reduction is depicted in Figure 3.1.

Namely, we exclude all the points on the lines L_{ij} such that neither L_{ij} nor $\bar{L}_{ij}$ contribute to cover any P_i

Theorem 3.2.1. *$Q \cup \{P_1, \ldots, P_m\}$ can be covered with $3m$ lines iff φ is satisfiable*

Proof. Since there are exactly m clauses, each contains 3 instance of variables, there are $3m$ instances of the variables in φ. If φ is satisfiable, we can use the following covering of lines. Let γ be a assignment that satisfies φ

- If v_i is true in γ, for each $v_i \in E_j$ we create a line L_{ij} that passes through P_j and all the points Q_{ij}.

- If v_i is false in γ, for each $\bar{v_i} \in E_j$ we create a line $\bar{L}_{ij}$ that passes through P_j and all the points $\bar{Q}_{ij}$.

Since there are $3m$ instances of the variables in φ, $3m$ lines covers all the points in $Q \cup \{P_1, \ldots, P_m\}$.

We prove the case where φ is not satisfiable by contradiction. Let us assume that φ is not satisfiable and $3m$ lines can cover $Q \cup \{P_1, \ldots, P_m\}$. Since $nm - 3m$ lines can cover $\{p^i_{kl}\} \setminus Q$. This implies a set of mn lines that covers the original instance of [MT82] corresponding to φ. Here we use the fact that in our instance, for each variable we must cover the corresponding sub grid in a unified direction (exactly as in [MT82]). This is in contradiction to the fact that φ is not satisfiable. □

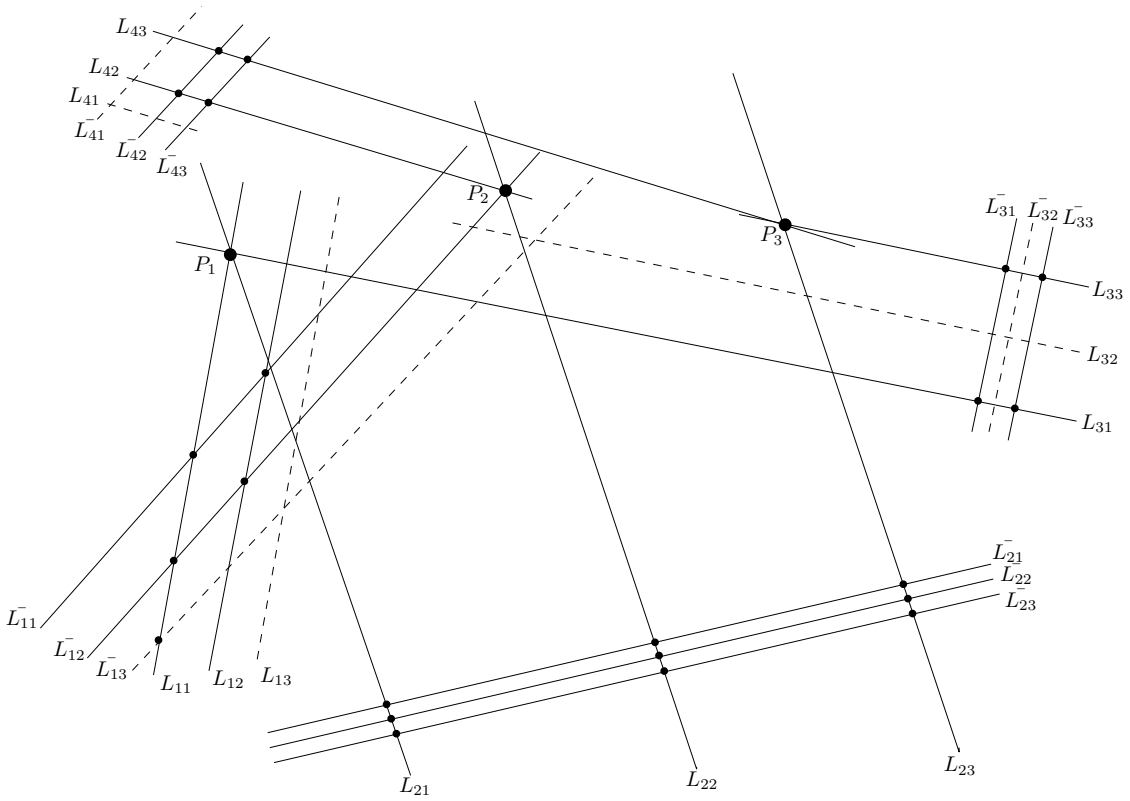

Figure 3.1: Example for $E_1 = v_1 \vee v_2 \vee v_3, E_2 = \bar{v_1} \vee v_2 \vee v_4, E_3 = v_2 \vee v_3 \vee v_4$. Each point P_i represent a clause E_i. With each variable v_i we associate a grid of points (in this case a 3×3 grid), such that if $v_i \in E_j$ then there exist a line L_{ij} passing through P_j and $\{p^i_{1j} \ldots p^i_{mj}\}$. Similarly if $\bar{v}_i \in E_j$ then there exist a corresponding line $\bar{L}_{ij}$ passing through P_j and $\{p^i_{j1} \ldots p^i_{jm}\}$. The dashed lines are lines that where constructed in the original reduction but are absence in our reductions.

Using the standard duality between points and lines in $\mathbb{R}^2$, [MT82] define a dual instance to the problem in which there are p lines and r points, and a point from the original instance lay on two lines iff the corresponding two dual points lay on the corresponding dual line.

Consider the dual problem to our enhanced reduction, referred to as the LC problem. In the corresponding dual instance we will have $O(n^2)$ lines and the optimum solution will consist of $3n$ points

Now apply k-median for lines on our LC instance with $k = 3n$. If the k-median value is 0, then this is a solution for the LC instance, implying that φ is satisfiable. Otherwise there is no solution to this instance, implying that φ is not satisfiable.

Let us assume that we can solve the k-median problem in time $f(n, k) \leqslant poly(n)2^{\epsilon k}$, then with the above reduction we can solve 3-SAT in time $f(n^2, 3n) \leqslant poly(n)2^{O(\epsilon n)}$

We note that our reduction also holds for any multiplicative approximation algorithm for k-median, since for optimal value 0 any multiplicative approximation will also be 0.

□

Chapter 4

k-median for lines: strong coreset

4.1 Coreset for k-median for lines

In this section we will present an algorithm that given n lines L in the plane returns a small *weighted* subset of lines L' such that for any k centers Φ it holds that the cost of clustering L with Φ is approximately that of clustering L'. Our algorithm will involve several steps as outlined below.

4.1.1 Outline

Let $L = \{l_1, \ldots, l_n\}$ be a set of lines in $\mathbb{R}^2$. For a set of k points $\Phi = \{\phi_1, \ldots, \phi_k\}$, let $Dist(\Phi, l_i)$ be the Euclidean distance between a line l_i and the closest point from Φ. We refer to the points of Φ as facilities. A line l is said to be served by $\phi \in \Phi$ if ϕ is the closest facility to l. Let $\varphi(\Phi, L) = \sum_{i=1}^{n} Dist(\Phi, l_i)$. In the k-median problem for lines one is to find a set of k points Φ^* such that $\varphi(\Phi, L)$ is minimized. We denote such Φ^* as the k-median points and $\varphi^*(L) = \varphi(\Phi^*, L)$ as the value of the k-median based solution. Let $L' = \{l_1, \ldots, l_m\}$. L' is defined as a strong coreset for the k-median problem for L, if for any set of k facilities Φ, $(1-\epsilon)\varphi(\Phi, L') \leqslant \varphi(\Phi, L) \leqslant (1+\epsilon)\varphi(\Phi, L')$.

In this section we present an algorithm, given L, that finds a strong coreset $L' \subseteq L$ of size $\frac{c^k}{\epsilon^{\frac{3k}{2}}} \log^{4k-4} n$ for some constant c. Our algorithm is based on an (α, β) bi-criteria approximation algorithm. In an (α, β) bi-criteria approximation, one seeks a set of points Φ' of relaxed side βk such that $\varphi(\Phi', L) \leqslant \alpha\varphi^*(L)$. Here α and β are typically greater then 1 and we refer to the output of an (α, β) bi-criteria algorithm as an (α, β) approximation solution.

Our algorithm for finding a coreset L' has 4 steps.

1. Find Φ', $|\Phi'| = O(k^2 \lg n \ln(\frac{k \lg n}{\epsilon})) = \beta k$ such that $\varphi(\Phi', L) \leqslant 8\varphi^*(L)$. Namely find an $(8, \beta)$ approximate solution Φ' for β of size approximately $\lg n$.

2. Using Φ', find Φ'', $|\Phi''| = O(\frac{k^2|\Phi|}{\epsilon} \lg n \ln(\frac{k \lg n}{\epsilon}) \log_{1+\epsilon} \frac{8n}{\epsilon})$ such that $\varphi(\Phi'', L) \leqslant 32\epsilon\varphi^*(L)$. Namely find a $(32\epsilon, \beta)$ approximate solution Φ' for β of size approximately $\lg^2 n$.

3. Move each line to its nearest facility in Φ''. This will create a partition of lines in L into subsets, each subset intersecting at a single point. We denote the geometrical structure of the subsets as "star shape".

4. Find a strong coreset for each star shaped subset. The union of the coresets will be the resulting coreset.

In general our algorithm works only in the plane $\mathbb{R}^2$. However, steps 1-3 work for higher dimensions also. We thus prove steps 1-3 in $\mathbb{R}^d$ and step 4 in $\mathbb{R}^2$.

4.1.2 Bi-criteria for k-median

The first phase of our algorithm finds a set of facilities Φ', $|\Phi'| \geqslant k$ such that $\varphi(\Phi', L) \leqslant 8\varphi^*(L)$. Our algorithm works in iterations. For a set of lines L, in each iteration we will show a way to select points to add to Φ' that cover a subset of lines in L. The union of the selected points in each iteration will be our facilities Φ'. Our algorithm is presented in Algorithm 3.

Lemma 4.1.1. *For a center $\phi \in \Phi$, let $L_\phi = \{l_1, \ldots, l_m\} \subseteq L$ be the set of lines that is being served by ϕ ordered by their distance from ϕ. For $l_j \in L_\phi$, let $proj_j$ be the projection of ϕ on l_j. Then for each line $l_h \in L_\phi$ such that $h \geqslant j$, $Dist(l_h, proj_j) \leqslant 2Dist(l_h, \phi)$.*

Proof. Let d be the distance between ϕ and $proj_j$, then for each line l_h, by the triangle inequality $Dist(l_h, proj_j) \leqslant Dist(l_h, \phi) + d$. Since for each line l_h such that $h \geqslant j$, $d \leqslant Dist(l_h, \phi)$, then $Dist(l_h, proj_j) \leqslant 2Dist(l_h, \phi)$. See Figure 4.1. □

Lemma 4.1.2. *For a center $\phi \in \Phi$, let $L_\phi = \{l_1, \ldots, l_m\} \subseteq L$ be the set of lines that is being served by ϕ ordered by their distance from ϕ. Let l_j be a line in L_ϕ and $proj_j$ be the projection of ϕ on l_j. For a line $l_h \in L_\phi$ let $proj_{j,h}$ be the closest point on l_j to l_h. Fix a line l_x in L_ϕ, and denote by d' the distance between $proj_j$ and $proj_{j,x}$. Then, for each line $l_h \in L_\phi$ such that $h \geqslant j$ and $Dist(proj_{j,h}, proj_j) \geqslant d'$ it holds that $Dist(l_h, proj_{j,x}) \leqslant 4Dist(l_h, \phi)$. See Figure 4.1.*

Proof. Since $Dist(proj_j, proj_{j,x}) \leqslant Dist(proj_j, proj_{j,h})$, then it can be seen that $Dist(l_h, proj_{j,x}) \leqslant 2Dist(l_h, proj_j)$ (this is immediate in $\mathbb{R}^2$ and follows from basic computations in higher dimensional space). By Lemma 4.1.1, for each line l_h such that $h \geqslant j$, $Dist(l_h, proj_j) \leqslant 2Dist(l_h, \phi)$, thus $Dist(l_h, proj_{j,x}) \leqslant 4Dist(l_h, \phi)$.

□

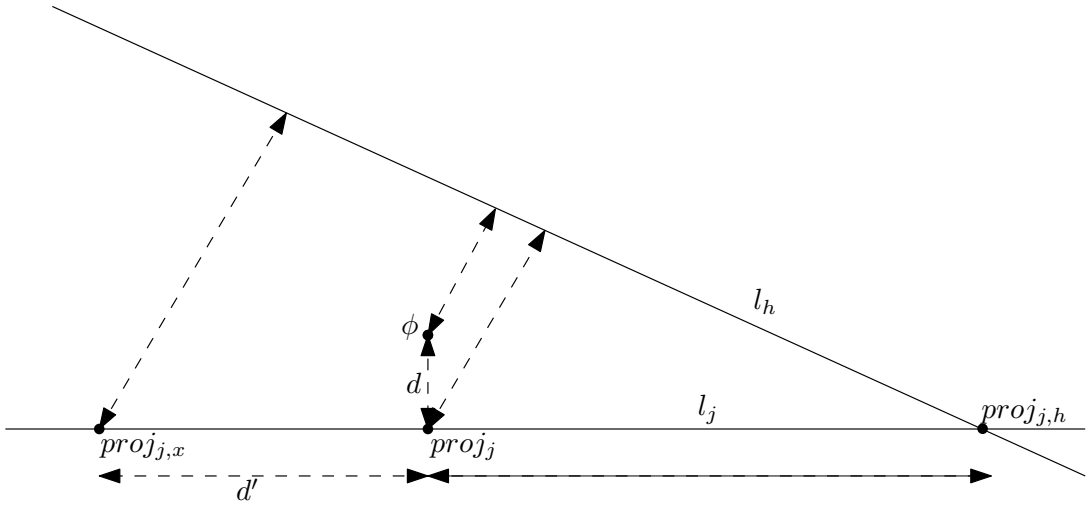

Figure 4.1: Example of definitions presented in Lemmas 4.1.1 and 4.1.2.

Lemma 4.1.3. *Let $\epsilon > 0$. If we set $\delta = \frac{1}{200}$, then with probability $1 - \epsilon$, Algorithm 3 finds a set Φ' such that $\varphi(\Phi', L) \leqslant 8\varphi^*(L)$, and $|\Phi'| = O(k^2 \lg n \ln(\frac{k \lg n}{\epsilon}))$.*

Algorithm 3 Bi-criteria for lines k-median(L, k, δ)

$\bar{L} \leftarrow L$
$n \leftarrow |L|$
$P \leftarrow \emptyset$
for $\lg n$ times **do**
  $\bar{n} \leftarrow |\bar{L}|$
  for $\frac{100^2k^2}{\delta^2} \ln(\frac{k \lg n}{\epsilon})$ times **do**
    Pick two lines l_j, l_x chosen randomly from $\bar{L}$ (in a uniform manner)
    $proj_{j,x} \leftarrow$ The closest point to l_x on l_j.
    $P \leftarrow P \cup proj_{j,x}$
  end for
  $\bar{L} \leftarrow \bar{L} \setminus$ the $\frac{\bar{n}}{2}$ closest lines to P
end for
$\Phi' \leftarrow P \cup$ a point on each of the remaining lines in $\bar{L}$
return Φ'

Proof. In order to analyze $\varphi(\Phi', L)$, we start with some terminology. Let n be the size of L and let $\bar{n}$ be the size of $\bar{L}$ (the remaining lines in each iteration). Let $proj_{jx}$ be the closest point on l_j to l_x. Let $\Phi = \{\phi_1, \ldots, \phi_k\}$ be a set of facilities, for a facility $\phi_i \in \Phi$, we define the cluster C_i as the set of lines from $\bar{L}$ that is being served by ϕ_i. We refer to $proj_{jx}$ as δ-good for a cluster C_i if:

- Both l_j and l_x are in C_i.
- l_j is among the $\delta|C_i|$ closest lines to ϕ_i in C_i.
- In the set $\{proj_{j,h} | l_h \in C_i\}$, $proj_{j,x}$ is among the $\delta|C_i|$ closest points to $proj_j$. Here $proj_j$ is the projection of ϕ_i on l_j.

In each iteration of our algorithm we choose $\frac{100^2k^2}{\delta^2} \ln \frac{k \lg n}{\epsilon}$ centers to be added to P. A line will be called good, if its distance to P is less then 4 times its distance to the optimal Φ, otherwise we will refer to it as bad. By Lemma 4.1.2, if $proj_{jx}$ is δ-good for C_i then there are at least $(1-2\delta)|C_i|$ good lines in C_i. Moreover a line will be called near if it is in the set of closest lines that are removed from $\bar{L}$ at the end of the iteration.

Let us analyze what happens in each iteration of Algorithm 3. We will start by analyzing the probability that after the iteration there are at least $(1-2\delta)(\bar{n} - \frac{\bar{n}(k-1)}{100k}) \geqslant (1-2\delta)\frac{99}{100}\bar{n}$ good lines in a single iteration. Here $\bar{n}$ is defined in each iteration of the algorithm.

For each C_i, let $n_i = |C_i|$. If $n_i \geqslant \frac{\bar{n}}{100k}$, then the probability that both l_j and l_x are from C_i is at least $\frac{1}{100^2k^2}$, and the probability that $proj_{jx}$ is δ-good for C_i is at least $\frac{\delta^2}{100^2k^2}$.

In each iteration we pick $\frac{100^2k^2}{\delta^2} \ln(\frac{k \lg n}{\epsilon})$ different centers (pairs of lines). Denote these centers by P.

Thus if $|C_i| \geqslant \frac{\bar{n}}{100k}$ then

$$\begin{aligned} Prob[\exists proj_{jx} \in P \text{ such that } proj_{jx} \textit{ is } \delta\text{-good for } C_i] &\geqslant 1 - \left(1 - \frac{\delta^2}{100^2k^2}\right)^{\frac{100^2k^2}{\delta^2} \ln(\frac{k \lg n}{\epsilon})} \\ &\geqslant 1 - \left(\frac{1}{e}\right)^{\ln(\frac{k \lg n}{\epsilon})} \\ &= 1 - \frac{\epsilon}{k \lg n} \end{aligned}$$

Thus, Prob[$\forall C_i$ s.t. $|C_i| \geqslant \frac{\bar{n}}{100k}$ $\exists proj_{jx} \in P$ s.t. $proj_{jx}$ is δ-good for C_i] $\geqslant 1 - \frac{\epsilon}{\lg n}$. Namely, after the first round, with probability $1 - \frac{\epsilon}{\lg n}$, there is a δ-good point in P for each cluster C_i that satisfies $|C_i| \geqslant \frac{\bar{n}}{100k}$. This implies that there are at least $(1-2\delta)\frac{99}{100}\bar{n}$ good lines in $\bar{L}$. We call such an iteration successful.

After $\lg n$ external iterations the probability that each iteration is successful is at least $1 - \frac{\epsilon}{\lg n}\lg n = 1 - \epsilon$ by the union bound.

Let us set $\delta = \frac{1}{200}$. Let $Near_i$ be the near lines in iteration i and let $Good_i$ be the good lines in iteration i. Let NG_i be the near lines that are good in iteration i, let NB_i be the near lines that are bad in iteration i, let FB_i be the far lines that are bad in iteration i and let FG_i be the far lines that are good in iteration i.

$$|Good_i| = \sum |C_i|(1-2\delta) \geqslant \frac{99}{100}\bar{n}(1-2\delta) \geqslant \frac{99}{100}\bar{n}\frac{99}{100} \geqslant \frac{9}{10}\bar{n}$$

This implies that $|NB_i| \leqslant \bar{n} - |Good_i| \leqslant \frac{1}{10}\bar{n}$. Similarly, $|FB_i| \leqslant \frac{\bar{n}}{10}$ thus $|FG_i| \geqslant \frac{\bar{n}}{2} - \frac{\bar{n}}{10} = \frac{4\bar{n}}{10}$. In iteration $i+1$ the new size of $\bar{L}$ is $\frac{\bar{n}}{2}$ and in $\bar{L}$ there will be at least $\frac{9}{10}\frac{\bar{n}}{2}$ good lines. Since $\frac{\bar{n}}{4}$ lines are near in iteration $i+1$ (namely chosen by the algorithm only in iteration $i+1$) and $\frac{\bar{n}}{4}$ are far, thus $|FG_i \wedge N_{i+1}| \geqslant \frac{4\bar{n}}{10} - \frac{\bar{n}}{4} = 0.15\bar{n} \geqslant |NB_i|$.

Now let us analyze the total cost of $\varphi(\Phi', L)$. We will do so by charging each of the removed lines in each iteration. In each iteration we remove all the near lines, however there could be bad lines in the set of those lines. Namely, lines l for which $\varphi(l, P)$ is larger then $4\varphi(l, \Phi)$. For each good and near line the cost of this line is at most 4 times its cost in the optimal solution. For each of the bad and near lines l_{bad} we will charge a line l_{good} in the next iteration with the following properties:

1. l_{good} is far and good in the current iteration.
2. l_{good} is near in the next iteration.
3. l_{good} has not been charged by any other line other then l_{bad}. This is possible since $|FG_i \wedge N_{i+1}| \leqslant |NB_i|$.

Let P_i be the set P after iteration i. It holds that

$$\begin{aligned}
\varphi(\Phi', L) &\leqslant \sum_i \sum_{l \in Near_i} Dist(P_i, l) \\
&= \sum_i \Big(\sum_{l \in NB_i} Dist(P_i, l) + \sum_{l \in NG_i} Dist(P_i, l) \Big) \\
&\leqslant \sum_i \sum_{l \in NB_i} Dist(P_i, l) + 4 \sum_i \sum_{l \in NG_i} Dist(\Phi, l) \\
&\leqslant \sum_i \sum_{l \in NB_i} Dist(P_i, l) + 4\varphi(\Phi, L)
\end{aligned}$$

Thus, we are left to analyze $\sum_i \sum_{l \in NB_i} Dist(P_i, l)$. We have shown that $|NB_i| \leqslant |FG_i \wedge N_{i+1}|$, thus for each $l_{bad} \in NB_i$ one can match a distinct line $l_{good} \in N_{i+1}$ such that

1. $Dist(l_{bad}, P_i) \leqslant Dist(l_{good}, P_i)$ (follows from the fact that $l_{good} \notin Near_i$ and $l_{bad} \in Near_i$).
2. $Dist(l_{good}, P_i) \leqslant 4Dist(l_{good}, \Phi)$ (follows from the fact that $l_{good} \in Good_i$).
3. l_{good} is removed from $\bar{L}$ in iteration $i+1$ ($l_{good} \in N_{i+1}$).

By 3 it follows that any line l_{good} corresponds to at most a single line l_{bad}. By 1 and 2 we have $Dist(l_{bad}, P_i) \leqslant 4Dist(l_{good}, \Phi)$.

Thus :

$$\sum_i \sum_{l_{bad} \in NB_i} Dist(l_{bad}, P_i) \leqslant \sum_{l_{good} \in L} 4Dist(l_{good}, \Phi) \leqslant 4\varphi(L, \Phi)$$

Notice that in the final iteration $\bar{L}$ is of constant size and we pick a point to add to P for each $l \in \bar{L}$. Thus there are no bad lines in this iteration.

This means that for each line we charged at most 8 times the cost in the optimal solution (four for itself and four for a near bad line). Therefore the total cost $\varphi(\Phi', L)$ is at most 8 times that of $\varphi(\Phi, L)$. Choosing Φ to be the optimal k centers, we have $\varphi(\Phi, L) \leqslant 8\varphi^*(L)$. Note that $|\Phi'| = O(k^2 \lg n \ln(\frac{k \lg n}{\epsilon}))$, as in line 13 of Algorithm 3 the size of the remaining $\bar{L}$ is at most constant.

□

Lemma 4.1.4. *Algorithm 3 has a running time of* $O(nk^2 \ln(\frac{k \lg n}{\epsilon}) \lg n)$.

Proof. The loop in line 4 is executed $\lg n$ times. The loop in line 6 is executed $\frac{100^2 k^2}{\delta^2} \ln(\frac{k \lg n}{\epsilon})$ times for each inner iteration, resulting in a total of $O(k^2 \ln(\frac{k \lg n}{\epsilon}) \lg n)$ times. Each step inside that takes time of $O(1)$. In line 11, finding the distance for each line in $\bar{L}$ can be done in total time $O(k^2 \ln(\frac{k \lg n}{\epsilon}) \lg n)$ for all the iterations, which yield a total time of $O(nk^2 \ln(\frac{k \lg n}{\epsilon}) \lg n)$. Thus the total running time of the algorithm is $O(nk^2 \ln(\frac{k \lg n}{\epsilon}) \lg n)$. □

4.1.3 ϵ net

Let $\epsilon > 0$. For the second part of our algorithm we would like to find a set of facilities Φ'' such that $\varphi(\Phi'', L) \leqslant \epsilon\varphi(\Phi', L)$. We will do this by selecting a set of facilities corresponding to each facility $\phi' \in \Phi'$. We will create the set of facilities for $\phi' \in \Phi'$ in the following way. Consider a ball centered around ϕ' with radius $r_1 = \frac{\epsilon\varphi(\Phi', L)}{n}$. Let $\bar{L}$ be the set of lines in L served by ϕ'. Now consider the set of balls B_i with radius $r_i = (1+\epsilon)r_{i-1}$ centered at ϕ'. Let $i = 1, \ldots, t$ where t is defined to be the smallest integer such that the balls of radius r_i centered at ϕ' cover (i.e., intersect) all the lines in $\bar{L}$. For each ball B_i above we add to Φ'' a set of facilities on the boundary of B_i. The set added "covers" the boundary in the sense that every point on the boundary of B_i is of distance at most ϵr_i from some added facility. As we are in $\mathbb{R}^2$ this set if of size $\frac{2\pi r_i}{\epsilon r_i} = \frac{2\pi}{\epsilon}$. (For the case of $\mathbb{R}^d$, such a set can be seen to be of size $(\frac{\sqrt{4\varepsilon\pi}}{\epsilon})^{d-1}$ [GLS10]).

Lemma 4.1.5. *The size of* Φ'' *is* $\frac{2\pi|\Phi'|}{\epsilon} \log_{1+\epsilon} \frac{n}{\epsilon} = O(\frac{k^2}{\epsilon} \lg n \ln(\frac{k \lg n}{\epsilon}) \log_{1+\epsilon} \frac{n}{\epsilon})$

Proof. Let t be the number of balls created for a facility ϕ'. We start by bounding t. The distance from ϕ' to the farthest line in $\bar{L}$ is at most $\varphi(\Phi', L)$. Thus, $r_1(1+\epsilon)^t \leqslant \varphi(\Phi', L)$, and $t \leqslant \log_{1+\epsilon} \frac{n\varphi(\Phi', L)}{\epsilon\varphi(\Phi', L)} = \log_{1+\epsilon} \frac{n}{\epsilon}$. The number of facilities created on each ball is $\frac{2\pi r}{\epsilon r} = \frac{2\pi}{\epsilon}$. We conclude that the total number of facilities for each ϕ' is $\frac{2\pi}{\epsilon} \log_{1+\epsilon} \frac{n}{\epsilon}$. Finally, the total number of facilities is $\frac{2\pi|\Phi'|}{\epsilon} \log_{1+\epsilon} \frac{n}{\epsilon}$ □

Lemma 4.1.6. $\varphi(\Phi'', L) \leqslant 4\epsilon\varphi(\Phi', L)$

Proof. Let $l \in L$ be a line of distance d from ϕ'. If $d \leqslant r_1$ then by construction $d \leqslant \frac{\epsilon\varphi(\Phi', L)}{n}$. If $d > r_1$, let i be the index such that $r_{i-1} < d < r_i$. It holds that $r_i - r_{i-1} \leqslant \epsilon r_{i-1}$ due to construction. As $d > r_{i-1}$ we conclude $r_i - d < r_i - r_{i-1} \leqslant \epsilon r_{i-1} < \epsilon d$.

Since the distance from each point of the ball B_i to the nearest facility is less then ϵr_i then $Dist(l,\Phi'') \leqslant \epsilon d + \epsilon r_i \leqslant \epsilon d + (1+\epsilon)\epsilon d \leqslant 3\epsilon Dist(l,\Phi')$. Thus

$$\begin{aligned}
\varphi(\Phi'',L) &= \sum_{l\in L} Dist(l,\Phi'') \\
&= \sum_{Dist(l,\Phi')\leqslant r_1} Dist(l,\Phi'') + \sum_{Dist(l,\Phi')>r_1} Dist(l,\Phi'') \\
&\leqslant n\frac{\epsilon\varphi(\Phi',L)}{n} + 3\sum_{Dist(l,\Phi'')>r_1} \epsilon Dist(l,\Phi') \\
&\leqslant \epsilon\varphi(\Phi',L) + 3\epsilon \sum_{Dist(l,\Phi'')>r_1} Dist(l,\Phi') \\
&\leqslant 4\epsilon\varphi(\Phi',L)
\end{aligned}$$

□

We now conclude this section by the following lemma.

Lemma 4.1.7. *Let L' be the set of lines resulting from moving each line in L to its nearest facility in Φ'' (by additive shifts), then for any set of facilities Φ: $|\varphi(\Phi,L) - \varphi(\Phi,L')| \leqslant 32\epsilon\varphi^*(L)$.*

Proof. In Lemma 4.1.6 we proved that $\varphi(\Phi'',L) \leqslant 4\epsilon\varphi(\Phi',L)$, we also proved in Lemma 4.1.3 that $\varphi(\Phi',L) \leqslant 8\varphi^*(L)$. Therefor $\varphi(\Phi'',L) \leqslant 32\epsilon\varphi^*(L)$. L' is created by moving each line to its closest facility. Namely for each set of facilities Φ, by the triangle inequality it holds that $|\varphi(\Phi,L) - \varphi(\Phi,L')| \leqslant 32\epsilon\varphi^*(L)$.

□

4.2 Strong coreset for k-median of lines

In the previous section we proved that one may assume with little loss, that our set of lines are partitioned into few groups, such that each group intersects at a common point (see Lemma 4.1.7). We now concentrate on one such group.

Let $L = \{l_1,\ldots,l_n\}$ be a set of n lines, all intersecting at a point p_C. Let C be a circle with center p_C and radius 1. We define the CK-median as k points $\Phi = \{\phi_1,\ldots,\phi_k\}$ outside C such that $\varphi(\Phi,L) = \sum_{i=1}^{n} Dist(\Phi,l_i)$ is minimized.

Lemma 4.2.1. *Let C_{up} be a half space defined by a line passing trough p_C and let $C_{down} = \mathbb{R}^2 \setminus C_{up}$. For a given set of facilities Φ we can create a set Φ' for which all of its points are in C_{up}, such that for any L, $Dist(\Phi,L) = Dist(\Phi',L)$.*

Proof. We can create Φ' in the following way. For each point $\phi \in \Phi$ that is in C_{down} with distance d_ϕ from p_C, we will look at a line l_ϕ that passes through ϕ and p_C. We will replace ϕ with a point ϕ' on that line on the other side of p_C ($l_\phi \cap C_{up}$) with distance d_ϕ from p_C.

Since each line $l \in L$ passes through p_C, and l_ϕ passes through p_C, the distance of each point p_ϕ on l_ϕ to l is $Dist(p_\phi,p_c)\sin\alpha$, where α is the angle between l and l_ϕ. Since the distance of ϕ and ϕ' from p_C is the same and they are lying on the same line l_ϕ, then $Dist(\phi,l) = Dist(\phi',l)$. We conclude that $Dist(\Phi,l) = Dist(\Phi',l)$. □

Definition 4.2.1. *Let $L = \{l_1 \dots l_n\}$ be a set of n lines. For a set of k points $\Phi = \{\phi_1, \dots, \phi_k\}$, with weights $\{w_{\phi_1}, \dots, w_{\phi_k}\}$, let the weighted Euclidean distance of l and ϕ_i be the Euclidean distance of l and ϕ_i multiplied by w_{ϕ_i}, and let $Dist(\Phi, l_i)$ be the minimum weighted Euclidean distance between a line l_i and the points from Φ. Let $\varphi(\Phi, L) = \sum Dist(\Phi, l)$.*

Lemma 4.2.2. *For a given set of facilities Φ outside C, we can create a set of weighted facilities Φ' such that all $\phi \in \Phi'$ are on C and $\varphi(\Phi, L) = \varphi(\Phi', L)$.*

Proof. For each $\phi \in \Phi$, let d_ϕ be the distance between ϕ and p_C (the center of C), and let l_ϕ be the line connecting ϕ and p_C. The distance from ϕ to a line $l \in L$ is $d_\phi \sin\alpha$, where α is the angle between l_ϕ and l. We now define the set of facilities Φ' such that for each $\phi \in \Phi$ we have a facility ϕ' with weight d_ϕ, located at the intersection point of l_ϕ and C. For each line $l \in L$ the weighted distance from ϕ' is $w(\phi')\sin\alpha = d_\phi \sin\alpha$, which is the weighted distance from ϕ. □

Definition 4.2.2. *Let $L = \{l_1 \dots l_n\}$ be a set of n lines, all intersecting at a point p_C, a CK-Median strong coreset is defined as a set of lines $L' \subseteq L$ of size ϵn for which, for each weighted facility set Φ of size k, it holds that, $(1-\epsilon)\varphi(\Phi, L') \leqslant \varphi(\Phi, L) \leqslant (1+\epsilon)\varphi(\Phi, L')$.*

4.2.1 $k = 1$: Finding Minimal Median for C1-Median

Lemma 4.2.3. *We can find the C1-Median in time $O(n\log(n))$.*

Proof. We first prove the following claim.

Claim 4.2.1. *The C1-median $\phi(C)$ is on the intersection of C and one of the lines.*

Proof. Lets assume in contradiction that the median point $\phi(C)$ in not on the intersection of C and one of the lines. As we allow only median centers that are outside C or on C, this also holds for $\phi(C)$. If $\phi(C)$ is not on any of the lines of L, we can create a new line l, that passes through $\phi(C)$ and does not intersect C. Unless all the lines in L are parallel to l (in this case $\phi(C)$ can be taken to be a point on one of the lines in L, and the median value is 0), l intersects another line from L. Let P_l be the set of intersection points of lines in L with the line l. Let $\varphi^*(L) = \sum_{l\in L} Dist(l, \phi(C))$ and for a point $p_l \in P_l$ let $\varphi(p_l, L) = \sum_{l\in L} Dist(l, p_l)$. By Claim 2.1.1, there exists a point $p_l \in P_l$ such that $\varphi^*(L) \geqslant \varphi(p_l, L)$, implying that $\varphi^*(L) = \varphi(p_l, L)$. Thus we may assume that $\phi(C)$ lie on the intersection of l and a line l' from L. Notice that p is outside C. Now consider the intersection point p of l' and C. It holds that $\varphi(p, L) \leqslant \varphi(\phi(C), L)$ as any line in L is closer to p than to $\phi(C)$. Thus, as before, we may assume that $\phi(C)$ equals p which contradicts our assumption. □

Since the minimal C1 median is on the intersection of C and one of the lines, we have n candidates for the minimal median. Thus, naively, one can compute the C1 median in time $O(n^2)$. In what follows, we present an alternative iterative algorithm which computes the C1 median in time $O(n\log(n))$.

Our algorithm will first sort the lines by their appearance on the cycle C, assume that the sorted lines are $l_1, \dots, l_n$ in clockwise order. We then scan the lines one by one, updating the value $\varphi(p)$ for each intersection point of a line $l \in L$ and C. We now show that scanning all the lines and calculating $\varphi(p)$ for each intersection point can be done dynamically in total running time of $O(n)$, which suffices to prove our assertion. To do so we will use Lemma B.0.3 of the appendix.

We, first calculate the angles $\alpha_1, \dots, \alpha_n$ between l_1 and the remaining lines $l_2, \dots, l_n$ (here, we set α_1 to be 0, and each α_i is at most π). Then we calculate $\sum_{l_i\in L}\sin\alpha_i$, $\sum_{l_i\in L}\cos\alpha_i$, $\sum_{l_i\in L(p_1)}\sin\alpha_i$ and $\sum_{l_i\in L(p_1)}\cos\alpha_i$. This can be done in time $O(n)$. Here p_1 is the intersection of l_1 and C; and $L(p_1)$ are the set of lines that lie in counter clockwise order to the left of p_1 with angle $\alpha_i < \pi/2$. Notice that these calculations include $\varphi(p_i, L) = \sum_{l_i\in L}\sin\alpha_i$. We can then calculate the

corresponding sums for l_2 using Lemma B.0.3 of the appendix in a running time of $O(|L_1| + |L_2|)$ where L_1 is the set of lines between l_1 and l_2, which is empty in our case, and L_2 is the set of lines between the lines perpendicular to l_1 and l_2. In general, Lemma B.0.3 allows us to compute the sums above for l_{i+1} once they are given for l_i in running time proportional to the number of lines between the lines perpendicular to l_i and l_{i+1} (denoted by L_2 in the lemma). As the lines appear in sorted order, each line can belong to the set L_2 only once, and thus the total running time is linear. After we calculate $\varphi(p_i, L)$ for all the intersection points, we then choose the minimum in time $O(n)$. This bring to a total running time of $O(n \lg n)$.

□

4.2.2 $k = 1$: Strong coreset for C1-Median

In this section we will present an algorithm that finds a strong coreset for C1-Median. This algorithm uses similar techniques to the ones defined in [HPK05]. In [HPK05], Har-Peled and Kushal present a technique for finding a strong coreset for the k-median problem for weighted points on a line, that finds a coreset of size $O(\frac{k}{\epsilon})$.

Lemma 4.2.4. *Let $L = \{l_1 \dots l_n\}$ be a set of lines intersecting at p_C. We can find a weighted set L' of size $O(\frac{1}{\epsilon})$ such that for each weighted facility ϕ, $(1 - c\epsilon^{\frac{2}{3}})\varphi(\phi, L') \leqslant \varphi(\phi, L) \leqslant (1 + c\epsilon^{\frac{2}{3}})\varphi(\phi, L')$. Here, c is a sufficiently large constant. The running time of our algorithm is $O(n \log(n))$.*

Proof. Note that by Lemmas 4.2.1 and 4.2.2 we can replace any set of facilities Φ with a weighted facility set Φ' laying on C_{up}. Let $L = \{l_1 \dots l_n\}$ be a set of lines intersecting at p_C, we start with some definitions depicted in Figure 4.2. Let $A \subseteq L$ be a set of lines in which the angle between each two lines is less then $\frac{\pi}{4}$. Let l_{left} be the leftmost (in counter clockwise order) line in A. For a line $l_i \in A$, let α_i be the angle between l_i and l_{left}. Let l_A be the line passing through P_C, with angle $\frac{\sum_{l_i \in A} \alpha_i}{|A|}$ to the right of l_{left}. We will also give l_A a weight $w(A) = |A|$. Let p_A be the intersection point of l_A and the circle C. Let us define $\xi^*(A) = \sum_{l \in A} Dist(l, p_A)$. Let V be the value of the 1-median for L on C, found by the C1-median algorithm from the previous section.

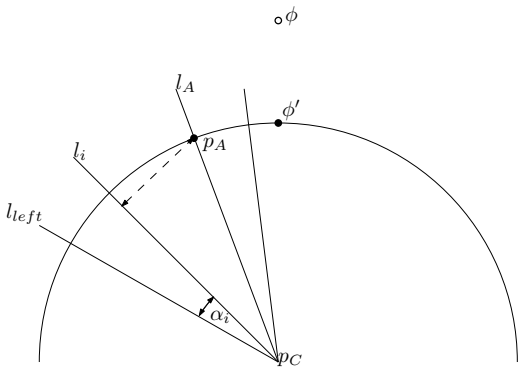

Figure 4.2: Example of lines in A as defined in Lemma 4.2.4

We now describe our algorithm. We will start by scanning the lines in $L = \{l_1, \dots, l_n\}$ by the direction of the intersection on C, and group them into batches. Namely we "grow" each batch A of lines in L sequentially until one of the following occur

- $\xi^*(A) = \epsilon V$. To ensure $\xi^*(A) = \epsilon V$ and not $\xi^*(A) > \epsilon V$, we will divide the last line in each batch into two weighted lines l_1, l_2 with total weight of 1. In this case $Dist(l_1, p_A) = w(l_1)Dist(l_1, p_A)$

- The angle between the start of the batch and the current scanned line $\theta \leqslant \epsilon$. If the angle is bigger then ϵ we will take the previous line.

Let $B = B_1, \ldots, B_u$ be the resulting batches, and let $L' = \{l_{B_1}, \ldots, l_{B_u}\}$. Let ϕ be a facility point outside C. We will scale ϕ, to ϕ' on C, and analyze the total error between L and L' on the facility ϕ'. By Lemma 4.2.2 this will suffice to prove our claim.

We now analyze the total error $\zeta = |\sum_{l_i \in L} Dist(l_i, \phi) - \sum_{B_i \in B} |B_i| \cdot Dist(l_{B_i}, \phi)| = d_\phi |\sum_{l_i \in L} Dist(l_i, \phi') - \sum_{B_i \in B} |B_i| \cdot Dist(l_{B_i}, \phi')|$.

Let us define $\xi(B_i) = |\sum_{l_i \in B_i} Dist(l_i, \phi') - |B_i| \cdot Dist(l_{B_i}, \phi')|$. It holds that $\zeta \leqslant d_\phi \sum_{B_i \in B} \xi(B_i)$. For a facility ϕ' and a batch B_i we will refer to ϕ' as inside B_i if ϕ' is to the left of the rightmost line in B_i and to the right to the leftmost line in B_i. We will divide the analysis into two different possibilities.

- Case 1: ϕ' is inside B_i
- Case 2: ϕ' is outside B_i

Consider B_i that contains ϕ', there could be at most two such batches. For such B_i, for each line $l \in B_i$, by Lemma B.0.2 $|Dist(l_{B_i}, \phi') - Dist(l, \phi')| \leqslant Dist(l, p_B)$ (p_B is the intersection point of l_{B_i} and C). Thus the total sum of error to $\xi(B_i) \leqslant \xi^*(B_i) = \epsilon V$.

We now consider B_i that doesn't contain ϕ'. Let us assume w.l.o.g that ϕ' is to the right of B_i. For these batches, define l'_ϕ as the line passing through ϕ' and p_C, β as the angle between l'_ϕ and the leftmost line in B_i, and for a line $l_j \in B_i$ define α_j as the angle between l_j and the leftmost line in B_i.

$\xi(B_i) = |\sum_{l_j \in B_i} Dist(l_i, \phi') - |B_i| \cdot Dist(l_{B_i}, \phi')| = |\sum_{l_j \in B_i} \sin(\beta - \alpha_j) - |B_i| \cdot \sin(\beta - \frac{\sum \alpha_j}{|B_i|})|$.

Claim 4.2.2. *For a batch B_i that doesn't contain ϕ' it holds that $\xi(B_i) \leqslant 6\epsilon^{\frac{2}{3}} \sum_{l_j \in B_i} \sin(\beta - \alpha_j)$.*

Proof. We divide our proof into two cases. The case where $\beta \in [0, 2\epsilon^{\frac{1}{3}}]$ and the case where $\beta \in [2\epsilon^{\frac{1}{3}}, \frac{\pi}{2}]$.

Let $\beta \in [0, 2\epsilon^{\frac{1}{3}}]$, in this case, the fact that the facility is close to B_i implies that the distance from the facility to the lines of B_i is close to the distance between the facilities and the points of $B_i \cap C$. We will use the following facts in our analysis: Since $\sin(x) = x - (\frac{x^3}{3!} - \frac{x^5}{5!}) - (\frac{x^7}{7!} - \frac{x^9}{9!}) - \ldots$ then $\sin(x) \leqslant x$, for $x \leqslant 1$. Since $\sin(x) = x - \frac{x^3}{3!} + (\frac{x^5}{5!} - \frac{x^7}{7!}) + (\frac{x^9}{9!} - \ldots$ then $\sin(x) \geqslant x - \frac{x^3}{3!}$, for $x \leqslant 1$. Thus, for the range $x \in [0, 2\epsilon^{\frac{1}{3}}]$, $x(1 - \frac{4\epsilon^{\frac{2}{3}}}{3!}) \leqslant \sin(x) \leqslant x$

If $\sum_{l_j \in B_i} \sin(\beta - \alpha_j) \geqslant |B_i| \cdot \sin(\beta - \frac{\sum \alpha_j}{|B_i|})$, then

$$\begin{aligned}
\xi(B_i) &= \sum_{l_j \in B_i} \sin(\beta - \alpha_j) - |B_i| \cdot \sin(\beta - \frac{\sum \alpha_j}{|B_i|}) \\
&\leqslant \sum_{l_j \in B_i} (\beta - \alpha_j) - |B_i| \cdot (1 - \frac{4\epsilon^{\frac{2}{3}}}{3!})(\beta - \frac{\sum \alpha_j}{|B_i|}) \\
&= |B_i|\beta - \sum \alpha_j - (1 - \frac{4\epsilon^{\frac{2}{3}}}{3!})(|B_i|\beta - \sum \alpha_j)
\end{aligned}$$

Otherwise,

$$\begin{aligned}\xi(B_i) &= |B_i| \cdot \sin(\beta - \frac{\sum \alpha_j}{|B_i|}) - \sum_{l_j \in B_i} \sin(\beta - \alpha_j) \\ &\leqslant |B_i|(\beta - \frac{\sum \alpha_j}{|B_i|}) - (1 - \frac{4\epsilon^{\frac{2}{3}}}{3!}) \sum_{l_j \in B_i} (\beta - \alpha_j) \\ &= |B_i|\beta - \sum \alpha_j - (1 - \frac{4\epsilon^{\frac{2}{3}}}{3!})(|B_i|\beta - \sum \alpha_j)\end{aligned}$$

Therefore

$$\begin{aligned}\xi(B_i) &\leqslant |B_i|\beta - \sum \alpha_j - (1 - \frac{4\epsilon^{\frac{2}{3}}}{3!})(|B_i|\beta - \sum \alpha_j) \\ &= |B_i|\beta - \sum \alpha_j - |B_i|\beta + \sum \alpha_j + \frac{4\epsilon^{\frac{2}{3}}}{6}(|B_i|\beta - \sum \alpha_j) \\ &= \frac{2\epsilon^{\frac{2}{3}}}{3}(|B_i|\beta - \sum \alpha_j) \\ &= \frac{2\epsilon^{\frac{2}{3}}}{3}(\sum(\beta - \alpha_j)) \\ &\leqslant \frac{2\epsilon^{\frac{2}{3}}}{3} \frac{\sum \sin(\beta - \alpha_j)}{1 - \frac{4\epsilon^{\frac{2}{3}}}{3!}} \\ &< \frac{6\epsilon^{\frac{2}{3}}}{3} \sum \sin(\beta - \alpha_j)\end{aligned}$$

Now let $\beta \in [2\epsilon^{\frac{1}{3}}, \frac{\pi}{2}]$. The fact that the facility is not too close to B_i causes the lost of each line to be relative to its distance. We will show that for each line $l_j \in B_i$, $|\sin(\beta - \alpha_j) - \sin(\beta - \frac{\sum \alpha_j}{|B_i|})| \leqslant \frac{3}{2}\epsilon^{\frac{1}{3}} \sin(\beta - \alpha_j)$. As $\beta \geqslant 2\epsilon^{\frac{1}{3}}$, and $\sin(x) > x - \frac{x^3}{3!}$, for all $\beta \in [2\epsilon^{\frac{1}{3}}, \frac{\pi}{2}]$, it holds that $\sin(\beta) \geqslant \sin(2\epsilon^{\frac{1}{3}}) \geqslant 2\epsilon^{\frac{1}{3}}(1 - \frac{4\epsilon^{\frac{2}{3}}}{3!})$. Since $\epsilon < 1$ and $\sin(\beta) > 2\epsilon^{\frac{1}{3}}(1 - \frac{4}{3!}) = \frac{4}{6}\epsilon^{\frac{1}{3}}$, we have that $\frac{3}{2}\epsilon^{\frac{2}{3}} \sin(\beta) > \epsilon$ for all $\beta \in [2\epsilon^{\frac{1}{3}}, \frac{\pi}{2}]$. Since $\beta \leqslant \frac{\pi}{2}$, it always holds that $|\sin\beta - \sin(\beta - \epsilon)| \leqslant \epsilon$. Thus, $|\sin\beta - \sin(\beta - \epsilon)| \leqslant \epsilon < \frac{3}{2}\epsilon^{\frac{2}{3}} \sin(\beta)$.

As $\sin(x) \geqslant \frac{x}{3}$ for $x \leqslant \frac{\pi}{2}$ and $\beta \geqslant 2\epsilon^{\frac{1}{3}}$, we have $\epsilon \leqslant \beta - \epsilon < 3\sin(\beta - \epsilon)$. As $\sin(\beta) - \sin(\beta - \epsilon) \leqslant \epsilon$, we also have $\sin(\beta) \leqslant \epsilon + \sin(\beta - \epsilon) \leqslant 4\sin(\beta - \epsilon)$. Here we have used that $\epsilon \leqslant 1$. Thus, $|\sin\beta - \sin(\beta - \epsilon)| \leqslant \frac{3}{2}\epsilon^{\frac{2}{3}} \sin(\beta) \leqslant \frac{12}{2}\epsilon^{\frac{2}{3}} \sin(\beta - \epsilon)$. Since both $\beta - \frac{\sum \alpha_j}{|B_i|} \leqslant \beta$, and $\alpha_j \leqslant \epsilon$, it holds that

$$\begin{aligned}|\sin(\beta - \alpha_j) - \sin(\beta - \frac{\sum \alpha_j}{|B_i|})| &\leqslant |\sin\beta - \sin(\beta - \epsilon)| \\ &\leqslant \frac{12}{2}\epsilon^{\frac{2}{3}} \sin(\beta - \epsilon) \\ &\leqslant 6\epsilon^{\frac{2}{3}} \sin(\beta - \alpha_j).\end{aligned}$$

$\xi(B_i) = \sum_{l_j \in B_i} |\sin(\beta - \alpha_j) - \sin(\beta - \frac{\sum \alpha_j}{|B_i|})| \leqslant 6\epsilon^{\frac{2}{3}} \sum_{l_j \in B_i} \sin(\beta - \alpha_j)$

□

We have shown that for B_i containing ϕ', $\xi(B_i) = O(\epsilon V)$, and for all other B_i, $\xi(B_i) = O(\epsilon^{\frac{2}{3}} \sum_{l_j \in B_i} Dist(l_i, \phi'))$. We conclude that the total error is $\zeta = O(d_\phi \epsilon^{\frac{2}{3}} \sum_{l_i \in L} Dist(l_i, \phi')) + \epsilon V = O(\epsilon^{\frac{2}{3}} V)$. Since $\varphi(\phi, L) \geqslant V$, thus $(1 - c\epsilon^{\frac{2}{3}})\varphi(\phi, L') \leqslant \varphi(\phi, L) \leqslant (1 + c\epsilon^{\frac{2}{3}})\varphi(\phi, L')$ for a sufficient large constant $c > 0$.

Now we will give a bound on the size of L'.

Claim 4.2.3. *The size of L' is $O(\frac{1}{\epsilon})$.*

Proof. We will start with some definitions. Let p be a point on the circle C. $Left(p, B_i)$ is defined as the lines in B_i to the left of p (in a counter clock wise order). $Right(p, B_i)$ is defined as the lines in B_i to the right of p (in a clock wise order). For a line $l \in B$, let $I(p, l)$ be the length of the arc from p to the intersection point of l and C.

We will start by showing that for each B_i, $\varphi^*(B_i) = \Omega(\xi^*(B_i))$ (here, $\varphi^*(B_i)$ is the optimal clustering of the batch B_i). The angle between l_{B_i} and the left most line in B_i equals $\frac{\sum \alpha_i}{|B_i|}$, thus

$$\sum_{l \in Left(p_{B_i}, B_i)} I(p_{B_i}, l) = \sum_{l \in Right(p_{B_i}, B_i)} I(p_{B_i}, l) = \frac{\xi^*(B_i)}{2}$$

$Dist(p_{B_i}, l) = \sin I(p_{B_i}, l)$. For $\epsilon \leqslant \frac{\pi}{4}$, $\frac{\sin \epsilon}{\epsilon} \geqslant \frac{\sin \frac{\pi}{4}}{\frac{\pi}{4}}$, thus $Dist(p_{B_i}, l) \geqslant \frac{\sin \frac{\pi}{4}}{\frac{\pi}{4}} I(p_{B_i}, l) \geqslant 0.9 I(p_{B_i}, l)$, thus

$$\sum_{l \in Left(p_{B_i}, B_i)} Dist(p_{B_i}, l) \geqslant 0.9 \frac{\xi^*(B_i)}{2}$$

Denote the optimal solution on B_i by ϕ_i and assume (w.l.o.g) that ϕ_i is to the right of p_{B_i}, then for each line $l \in Left(p_{B_i}, B_i)$, $Dist(\phi_i, l) \geqslant Dist(p_{B_i}, l)$, thus $\varphi^*(B_i) = \varphi(\phi_i, B_i) \geqslant \varphi(\phi_i, Left(p_{B_i}, B_i)) \geqslant \varphi(p_{B_i}, Left(p_{B_i}, B_i)) \geqslant 0.9 \frac{\xi^*(B_i)}{2}$

Consider a batch B_i such that $\xi^*(B_i) < \epsilon V$. Such batches are defined by their angle (between the leftmost and the rightmost line in the batch). Specifically for such batches B_i the angle between l_{left} of B_i and l_{left} of B_{i+1} is less than then ϵ. Thus the number of such batches is bounded by $\frac{\pi}{\epsilon} + 1$. The addition of one in $\frac{\pi}{\epsilon} + 1$ is attributed to the last batch B_u which may also have $\xi^*(B_u) < \epsilon V$.

For the rest of the batches, $\xi^*(B_i) = \epsilon V$, thus

$$V = \varphi^*(L) \geqslant \sum_{B_i \in B} \varphi^*(B_i) \geqslant 0.9 \sum \frac{\xi^*(B_i)}{2} \geqslant 0.9(|B| - \frac{\pi}{\epsilon} - 1)\frac{\epsilon V}{2}$$

Therefor $|B| \leqslant \frac{1}{0.9} \frac{2}{\epsilon} + \frac{\pi}{\epsilon} + 1 = O(\frac{1}{\epsilon})$

□

Claim 4.2.4. *The running time of our algorithm is $O(n \lg n)$.*

Proof. As our algorithm scans the lines, the lines need to be sorted, which can be done in time $O(n \lg n)$. We then turn to computing the batches. Namely, we scan the lines (in clockwise order) and compare $\xi^*(A) = \sum_{l \in A} Dist(l, p_A)$ with ϵV. The point p_A can be adjusted with each line added to A in constant time $O(1)$. Computing the value of $\xi^*(A)$ iteratively with each update can be done in a total running time of $O(n)$ using the techniques described in Lemma B.0.3 of the appendix.

Specifically, let A be the current batch, let A_i be the lines in iteration i, and let p_{A_i} and l_{A_i} be the values of p_A and l_A for A_i. Let $\alpha_{i1}, \ldots, \alpha_{ik}$, $k = |A_i|$, be the angles between l_{A_i} and the lines in

A_i. Our algorithm maintains the values $\xi^*(A_i) = \sum_{l_j \in A_i} \sin\alpha_{i,j}$, $\sum_{l_j \in A_i} \cos\alpha_{i,j}$, $\sum_{l_j \in L(p_{A_i})} \sin\alpha_{i,j}$ and $\sum_{l_j \in L(p_{A_i})} \cos\alpha_{i,j}$. Here $L(p_{A_i})$ is the set of lines in A_i that lie to the left of p_{A_i} (in counter clockwise order) with angle $\alpha_{i,j} < \pi/2$. Let $\bar{l}$ be the line added in iteration $i+1$, Using Lemma B.0.3, we now calculate $l_{A_{i+1}}$ and the above corresponding sums for A_{i+1} (notice that we also need to add the contribution of $\bar{l}$ after the use of Lemma B.0.3). Applying Lemma B.0.3 takes a running time of $O(|L_1| + |L_2|)$ where L_1 is the number of lines between l_{A_i} and $l_{A_{i+1}}$, and L_2 is the number of lines between the lines perpendicular to l_{A_i} and $l_{A_{i+1}}$. Since in our iterative process, the lines always progress in a sorted manner, each line will be counted a constant number of times. The total running time of the entire algorithm is thus $O(n \lg n)$.

□

This concludes that proof of Lemma 4.2.4. □

4.2.3 CK-Median strong coreset algorithm

We now present an algorithm that finds a strong coreset for CK-Median with $k > 1$. Our algorithm uses a technique similar to the one in [FFS06]. In [FFS06], Feldman, Fiat and Sharir describe a technique for finding a strong coreset for the k-median problem of size $O(log^{2k-1}n)$ for points on a line. Their algorithm addresses the case in which the facilities (centers), and not only the input points, may be weighted.

We will start with some definitions. For a set of weighted facilities Φ and a circle C, we define a Voronoi arc, as an arc on C such that all the points in this arc are served by the same facility, and the adjutant points of this arc (from both sides) are served by different facility. Later we will prove that the number of Voronoi arcs is bounded by $2k-1$.

Our algorithm appears in Algorithm 4, divide C to 8 equal parts $arc_i (i = 1 \dots 8)$, applying a subroutine algorithm V-Coreset specified in Algorithm 5 on each part. The union of the resulting coresets will be the final coreset.

Algorithm 5 gets a set of lines L, a number k and $0 < \epsilon \leqslant 1$. The Algorithm returns a set of lines L' such that for every set of weighted facilities Φ that creates at most k Voronoi arcs $|\varphi(\Phi, L) - \varphi(\Phi, L')| \leqslant \epsilon\varphi(\Phi, L)$. In order to simplify things, instead of looking at the number of facilities, we will look at the number of Voronoi arcs created by the facilities.

The main idea of Algorithm 5 is to divide the set of points to small groups and recursively find a strong coreset for each one of those groups. For the base case where there is single Voronoi arc, namely there is a single facility, we will use the $C1$-Median algorithm.

Our algorithm 5 uses the following notation. We use the distance function $I(p_1, p_2)$ for two points, as the length of the arc that passes through them. We denote the union of the intersection points of the lines in L with C as P. We then order the points by their appearance on C and split them in the middle to two sets S_L and S_R. We divide each one of the sets S_L and S_R to smaller sets B_i and in turn divide each B_i to smaller sets B_{ij}. Our scheme for dividing S_L, S_R and B_i are specified in Algorithm 5. We denote by Z the collection of all the sets B_{ij} in S. For algorithm 5 we will use a constant δ, later on we will bound δ.

Algorithm 4 CK-median for lines

```
1: divide C to 8 equals part Arc = /arc_1 ... arc_8/.
2: for each arc_i ∈ Arc ∩ C_up do
3:    L_i ← the lines from L that intersect arc_i
4:    S ← S∪ V-Coreset(L, 2k − 1, ε))
5: end for
6: return S
```

Algorithm 5 V-Coreset(L, k, ϵ)

if k=1 **then**
  return call to algorithm C1-Median (L, ϵ)
end if
$Z \leftarrow \emptyset$
$P \leftarrow L \cap C$
$n \leftarrow |P|$
$P_{mid} \leftarrow$ the middle point of P (when the points in P are ordered by their appearance on C).
let S_L be the points in P left to P_{mid} and let S_R be the points in P right to P_{mid}.
for each $S \in \{S_L, S_R\}$ **do**
  $I(S) \leftarrow$ The length of the arc containing S.
  Divide S to arcs B_i of length $\frac{2^{i-1}|I(S)|}{n^2}$, i is an index that runs from 1 to $\lceil \lg(n^2) \rceil$
  for each B_i **do**
    divide B_i into subsets B_{ij} (referred to as "intervals") in the following way. the first $\lceil \frac{\delta}{\epsilon} \rceil$ intervals contain a single point, the next $\lceil \frac{\delta}{\epsilon} \rceil$ intervals contain 2 points, the next $\lceil \frac{\delta}{\epsilon} \rceil$ intervals contain 4 points, and so on until we cover all of the points. For B_{1j} the division will start from the side furthest to the middle point, and for the remaining B_{ij} from the side closest to the middle point. See Figure 4.3.
    $Z \leftarrow Z \cup \{B_{ij}\}$
  end for
  $S_l \leftarrow \emptyset$ (for the right group we will use S_r)
  for each $B \in Z$ **do**
    $L_B \leftarrow$ lines that correspond the points in B
    $S_B \leftarrow$ V-Coreset$(L_B, k-1, \epsilon)$
    $S_l \leftarrow S_l \cup S_B$ (for the right group we will use S_r)
  end for
end for
$L' \leftarrow S_l \cup S_r$ with their weights
return L'

We start by bounding the size of our coreset:

Lemma 4.2.5. *Let k be the number of Voronoi cells. The number of lines in $S_r \cup S_l$ of Algorithm 5 is at most $\frac{c^k}{\epsilon^k} \log^{2k-2} n$ for some large enough constant c.*

Proof. We start by bounding the number of sets B_{ij} in Z. There are $O(\lg n)$ sets B_i by construction. For each set B_i, we start with sets B_{ij} of one point and after $\lceil \frac{\delta}{\epsilon} \rceil$ sets we multiply the number of points in the set by 2 for another $\lceil \frac{\delta}{\epsilon} \rceil$ sets. Using the fact that each B_i contains at most n points, it is not hard to verify that the number of sets in B_i is at most $\frac{\delta}{\epsilon} \lg(\frac{n\delta}{\epsilon}) = O(\epsilon^{-1} \log n)$. With $O(\lg n)$ sets B_i, $Z = O(\epsilon^{-1} \log^2 n)$. Thus there is a constant c_1 such that $Z \leqslant c_1(\epsilon^{-1} \log^2 n)$

Now we continue with the proof on the number of lines in $S_r \cup S_l$ defined in Algorithm 5. The proof is by induction on the number of Voronoi cells k. Let $T(k, \epsilon, n)$ be the maximum size of $S_r \cup S_l$ for a given ϵ and n. Let us assume that $T(k, \epsilon, n) \leqslant \frac{c^k}{\epsilon^k} \log^{2k-2} n$ for some constant c. For the base case $k = 1$, a single Voronoi cell can be created only by a single facility, $T(1, \epsilon, n) = O(\frac{1}{\epsilon})$ by Claim 4.2.3. Thus there is a constant c_2 such that $T(1, \epsilon, n) \leqslant c_2(\frac{1}{\epsilon})$ Let c be a constant bigger or equal to c_1 and c_2. $T(k, \epsilon, n) \leqslant |Z| \cdot T(k-1, \epsilon, n)$, by the assumption of the induction $T(k-1, \epsilon, n) \leqslant \frac{c^{k-1}}{\epsilon^{k-1}} \log^{2k-4}$. Thus

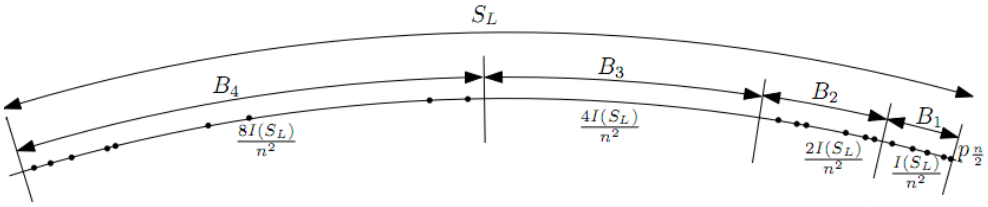

Figure 4.3: Example of partition of $I(S_L)$ to $B_1, \ldots, B_{\lceil \lg n^2 \rceil}$.

$$\begin{aligned} T(k,\epsilon,n) &\leqslant |Z| \cdot T(k-1,\epsilon,n) \\ &= c(\epsilon^{-1}\log^2 n) \cdot T(k-1,\epsilon,n) \\ &= c(\frac{1}{\epsilon}\log^2 n) \cdot \frac{c^{k-1}}{\epsilon^{k-1}}\log^{2k-4} n \\ &= \frac{c^k}{\epsilon^k}\log^{2k-2} n \end{aligned}$$

□

As seen in Lemma 4.2.2, for any set of facilities in $\mathbb{R}^2$ we can create a set of weighted facilities all laying on C_{up}. We will continue with the proof assuming all the facilities are on C_{up}.

Lemma 4.2.6. *Let arc_i be as defined in Algorithm 4. The number of Voronoi arcs created on each arc_i by the k weighted facilities is bounded by $2k-1$.*

Proof. In order to be able to use the V-Coreset algorithm we need to bound the number of Voronoi arcs generated by the weighted facilities on each arc_i. We consider Davenport-Schinzel sequences to bound the number of Voronoi arcs created on each arc_i. For a sequence $U = \{u_1, \ldots, u_m\}$ of integers, and two positive numbers k and s, a Davenport-Schinzel sequences $DS(k,s)$ is a sequence satisfying the following:

1. for each $u_i \in U$, $1 \leqslant u_i \leqslant k$
2. $u_i \neq u_{i+1}$ for each $i < m$.
3. there does not exist $s+2$ indices $1 \leqslant i_1 \leqslant i_2 \leqslant \ldots \leqslant i_{s+2} \leqslant m$ such that $u_{i_1} = u_{i_3} = \cdots = a$ and $u_{i_2} = u_{i_4} = \cdots = b$ and $a \neq b$

According to [SA96] if $s = 2$ then $DS(k,2) = 2k-1$.

Let U be the sequence of the Voronoi arcs created on arc_i. Each facility $\phi \in \{\phi_1, \ldots, \phi_k\}$ can create several intervals on arc_i, as the facilities are weighted. For each arc $u_i \in U$, the value of u_i will be the index of the facility that is associate with the arc. Condition 1 and 2 above can be verified due to our construction. In order to satisfy the third condition, we will show that for each two facilities, the weighted distance functions from a point on arc_i to each one of those facilities intersect at most twice.

Consider the arc length from a point on the circle to closest weighted facility on the circle. The arc distance of each facility $\phi \in \Phi$ to a point p on the circle is $w(\phi)\sin\theta$, where θ is the angle between the line (p_C, p) and the line (p_C, ϕ) (p_C is the center of the circle).

In order to find the distances for each point on the circle to its closest facility, we should look at the lower envelope of the functions $w(\phi_i)\sin\theta_i$ created by each facility. Since we only need to find a coreset for each arc_i at a time, we consider the number of times the functions intersect when θ_i is in an interval of length $\frac{\pi}{4}$.

It can be easily verified that any two functions of type $w(\phi_i)\sin\theta_i$ intersect at most twice in an interval of length $\frac{\pi}{4}$.

Since each two functions intersect at most twice there do not exist 4 indices $1 \leqslant i_1 \leqslant i_2 \leqslant i_3 \leqslant i_4 \leqslant m$ such that $u_{i_1} = u_{i_3} = a$ and $u_{i_2} = u_{i_4} = b$ and $a \neq b$. □

We now state and prove the main theorem of this section.

Theorem 4.2.1. *Let L be a set of n lines intersecting arc_i. Algorithm V-Coreset(L, k, ϵ) finds in time $O(n\frac{c^k}{\epsilon^k}\log^{2k-1} n)$, a strong coreset L' for L such that $|\varphi(L') - \varphi(L)| \leqslant O(\epsilon^{\frac{2}{3}}\varphi(L))$ of size $O(\frac{c^k}{\epsilon^k}\log^{2k-2} n)$ for some constant c.*

Proof. The size of the coreset is proved by Lemma 4.2.5. For an arc arc_i let arc_{i+1} be the adjacent arc in clockwise direction and arc_{i-1} be the adjacent arc in counter clockwise direction. Let $\rho = arc_i \cup arc_{i+1} \cup arc_{i-1}$.

Let Φ be a set of facilities. In Algorithm 5 for each batch $B \in Z$ we define a coreset S_B. Let $\varphi(B) = \sum_{l\in B} Dist(\Phi, l)$ be the sum of values of all the distances of lines in L intersecting B to their closest facility, and let $\varphi(S_B) = \sum_{l\in S_B} w(l)Dist(\Phi, l)$ be the weighted value of all the coreset lines intersecting B to their closest facility.

We will use the following observation which follows directly by our definition. We assume that in line 8 of Algorithm 5 we are considering the case $S = S_L$. Otherwise our analysis is analogous.

Observation 4.2.1.

1. *The length of the arc $I(B_i)$, for $i > 1$, is less then twice the length of all the intervals to its right, Namely $I(B_i) \leqslant 2\sum_{j=1}^{i-1} I(B_j)$.*

2. *For any $B_{ij} \in Z$ that contains at least two points, we have $|B_{ij}| \leqslant \frac{2\epsilon}{\delta}\sum_{m<j} |B_{im}|$.*

Let L' be the set of lines returned from algorithm 5. We will prove by induction that Algorithm V-Coreset(L, k, ϵ) return a set of lines L' such that for each set of weighted facilities Φ that create at most k Voronoi arcs $|\varphi(\Phi, L) - \varphi(\Phi, L')| \leqslant c\epsilon^{\frac{2}{3}}\varphi(\Phi, L)$ for sufficiently large c. We will assume that the induction is correct for V-Coreset$(L, k-1, \epsilon)$. A single Voronoi arc can be created only by a single facility, thus the base case of our induction is covered by Lemma 4.2.4.

If for a every $B \in Z$, the interval $I(B)$ intersects less than k Voronoi arcs, then by the induction assumption each recursive call to V-Coreset$(L, k-1, \epsilon)$ returns a set S_B such that $|\varphi(B) - \varphi(S_B)| \leqslant \epsilon^{\frac{2}{3}}\varphi(B)$, thus

$$|\varphi(L) - \varphi(L')| \leqslant \sum_{B\in Z} |\varphi(B) - \varphi(S_B)| \leqslant \sum_{B\in Z} \epsilon^{\frac{2}{3}}\varphi(B) = \epsilon^{\frac{2}{3}}\varphi(L)$$

Here, for a set X, $\varphi(X) = \varphi(\Phi, X)$.

Notice that there can be at most a single batch B such that the interval $I(B)$ intersects k Voronoi arcs. If there exist a batch $B \in Z$, such that the interval $I(B)$ intersects k Voronoi arcs, then

$$\begin{aligned}|\varphi(L)-\varphi(L')| &\leqslant \sum_{X\in Z}|\varphi(X)-\varphi(S_X)| \\ &= \sum_{X\in Z\setminus B}|\varphi(X)-\varphi(S_X)|+|\varphi(B)-\varphi(S_B)| \\ &\leqslant \sum_{X\in Z\setminus B}\epsilon^{\frac{2}{3}}\varphi(X)+|\varphi(B)-\varphi(S_B)|\end{aligned}$$

So we are left to proof that for the single batch B such that the interval $I(B)$ intersects k Voronoi arcs, $|\varphi(\Phi,B)-\varphi(\Phi,S_B)|\leqslant\epsilon^{\frac{2}{3}}\varphi(L)$.

For any facility $\phi\in\Phi$, since S_B was constructed recursively by $C1-coresets$, S_B is a 1 coreset for B, hence $|\varphi(\phi,B)-\varphi(\phi,S_B)|\leqslant\epsilon^{\frac{2}{3}}\varphi(\phi,B)$, so $\varphi(\phi,S_B)\leqslant(1+\epsilon^{\frac{2}{3}})\varphi(\phi,B)$. Therefore $|\varphi(\Phi,B)-\varphi(\Phi,S_B)|\leqslant\varphi(\Phi,B)+\varphi(\Phi,S_B)\leqslant\varphi(\phi,B)+\varphi(\phi,S_B)\leqslant(2+\epsilon^{\frac{2}{3}})\varphi(\phi,B)$.

Bounding $\varphi(\phi,B)$ by $\epsilon\varphi(\Phi,L)$: We will now bound $\varphi(\phi,B)$ by $\epsilon\varphi(\Phi,L)$. We will prove our claim for S_L, a similar proof can be applied to S_R.

For a batch $B_{ij}\in Z$, let B_L be the set of lines of B_i that lies to the left of B_{ij} and let B_R be the set of lines of B_i that lies to the right of B_{ij}. Let P_L be the set of lines of L that lies to the left of B_{ij} and let P_R be the set of lines of L that lies to the right of B_{ij}. Since B is the single batch such that the interval $I(B)$ intersects k Voronoi arcs, all the other batches in Z intersects a single Voronoi arcs. Therefor for the facility $\phi\in\Phi$ that creates the leftmost Voronoi arc, and a batch $X\in P_L$, $\varphi(\phi,X)=\varphi(\Phi,X)$. Thus $\varphi(\phi,P_L)=\varphi(\Phi,P_L)\leqslant\varphi(\Phi,L)$. Symmetrically for the facility $\phi\in\Phi$ that creates the rightmost Voronoi arc, and for a batch $X\in P_R$, $\varphi(\phi,X)=\varphi(\Phi,X)$. Thus $\varphi(\phi,P_R)=\varphi(\Phi,P_R)\leqslant\varphi(\Phi,L)$. Thus it is enough to proof that either $\varphi(\phi,B)\leqslant\epsilon\varphi(\phi,P_L)$ or $\varphi(\phi,B)\leqslant\epsilon\varphi(\phi,P_R)$.

We will divide our proof to the case when there is at least one facility not in ρ and the case when all the facilities are in ρ. We start with the case in which there is at least one facility not in ρ. We will assume w.l.o.g. that ϕ is to the left of B. Recall that $\rho=arc_{i-1}\cup arc_i\cup arc_{i+1}$.

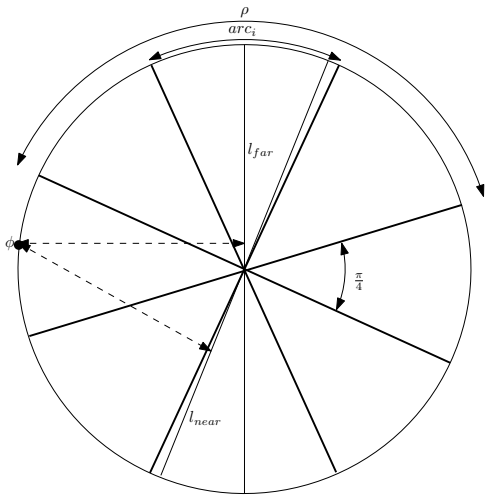

Figure 4.4: The distance from ϕ on $C_{up}-\rho$ to the nearest and farthest line on B.

Lemma 4.2.7. *Let ϕ be a facility on $C_{up}-\rho$, then $\varphi(\phi,B)\leqslant\epsilon\varphi(\phi,B_L)$.*

Proof. Let $Dist(\phi, l)$ be the distance from ϕ to any line that intersects arc_i, then it is not hard to verify that $\sin\frac{\pi}{4} \leqslant Dist(\phi, l) \leqslant 1$. Let l_{far} be the farthest line from ϕ in B and l_{near} be the nearest line to ϕ in B, thus $Dist(\phi, l_{far}) \leqslant \frac{1}{\sin\frac{\pi}{4}} Dist(\phi, l_{near})$. See Figure 4.4.

By observation 4.2.1(ii) $|B| \leqslant \frac{2\epsilon}{\delta}|B_L|$, thus $\varphi(\phi, B) \leqslant |B|Dist(\phi, l_{far}) \leqslant \frac{2\epsilon}{\delta}|B_L|Dist(\phi, l_{far}) \leqslant \frac{2\epsilon}{\delta}|B_L|Dist(\phi, l_{near})\frac{1}{\sin\frac{\pi}{4}} \leqslant \epsilon\varphi(\phi, B_L)\frac{2}{\delta\sin\frac{\pi}{4}}$. If we choose δ that is large enough so that $\frac{2}{\delta\sin\frac{\pi}{4}} \leqslant 1$ then $\varphi(B) \leqslant \epsilon\varphi(\phi, B_L)$. □

To summer up the case where ϕ is a facility on $C_{up} - \rho$. As ϕ is to the left of B then the leftmost Voronoi arc is created by ϕ, thus for a facility on $C_{up} - \rho$, $|\varphi(\Phi, B) - \varphi(\Phi, S_B)| \leqslant (2+\epsilon)\varphi(\phi, B) \leqslant (2+\epsilon)\epsilon\varphi(\phi, B_L)$. Recall that the latter is at most $(2+\epsilon)\epsilon\varphi(\Phi, B_L) \leqslant (2+\epsilon)\epsilon\varphi(\Phi, L)$.

We are left with the case that all of the facilities are in ρ. The proof in this case is very similar to the one in [FFS06]. This is due to the fact that for a facility in ρ and two lines l_1 and l_2 intersecting arc_i at p_1 and p_2, $I(\phi, p_1) \leqslant I(\phi, p_2)$ iff $Dist(\phi, l_1) \leqslant Dist(\phi, l_2)$. We show the complete proof.

Lemma 4.2.8. *For a set $B = B_{ij} \in Z$, where $|B| > 1$. Let P_L and P_R be the set of points that lie to the left and to the right of B respectively. Then with appropriate choice of $\delta \geqslant \frac{8}{\sin\frac{\pi}{4}}$, for any facility $\phi \in \Phi$ we have,*

(i) for $i = 1, \varphi(\phi, B) \leqslant \epsilon\varphi(\phi, P_L)$
(ii) for $i > 1, \varphi(\phi, B) \leqslant \epsilon\varphi(\phi, P_R)$

Proof. We use the following definition in our proof.

Definition 4.2.3. *For two points p_1, p_2 on ρ,*

- *$p_1 \leqslant p_2$ iff p_1 is to the left of p_2 on ρ.*
- *Let $I(p_1, p_2)$ be the length of the arc that starts at p_1 and ends at p_2 (in clockwise direction).*

We start with the following fact.

Fact 4.2.1. *Let l_1 and l_2 be lines passing trough the center of the circle C and let p_1 and p_2 be the intersection point of the lines and C. Let $0 \leqslant \alpha \leqslant \frac{\pi}{2}$ be the angle between l_1 and l_2, then $I(p_1, p_2)\sin\alpha = \alpha Dist(l_1, p_2) = \alpha Dist(l_2, p_1)$.*

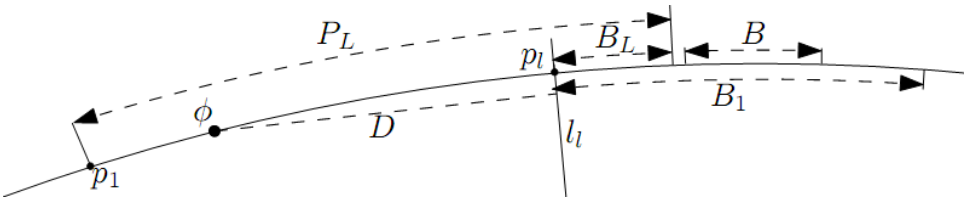

Figure 4.5: Proof of Lemma 4.2.8, case (i).

(i) If ϕ lies to the right of B, let B_L be the set of lines of B_1 that lies to the left of B. By observation 4.2.1 (ii), $|B| \leqslant \frac{2\epsilon}{\delta}|B_L|$. ϕ lies to the right of B therefore the distance from ϕ to each one of the points in B_L is greater than the distance from ϕ to each one of the points in B. Thus $\varphi(\phi, B) \leqslant \frac{2\epsilon}{\delta}\varphi(\phi, B_L) \leqslant \frac{2\epsilon}{\delta}\varphi(\phi, P_L)$.

Otherwise, if ϕ lies to the left of B or inside B, let $D = Dist(\phi, B_1)$ denote the distance between ϕ and the nearest line in B_1. Since $B \subseteq B_1$, and for every line l intersecting B, $Dist(\phi, l) < D + I(B_1)$, we have $\varphi(\phi, B) \leqslant |B|(D + I(B_1))$. Let B_L be the set of lines of B_1 that lies to

the left of B. By observation 4.2.1 (ii) $|B| \leqslant \frac{2\epsilon}{\delta}|B_L|$. Therefore, $\varphi(\phi, B) \leqslant |B|(D + I(B_1)) \leqslant \frac{2\epsilon}{\delta}|B_L|(D + I(B_1)) = \frac{2\epsilon}{\delta}|B_L|D + \frac{2\epsilon}{\delta}|B_L|I(B_1)$. See Figure 4.5.

Let us look at the first term, since D is at most the distance to the nearest line in B_L, $|B_L|D \leqslant \varphi(\phi, B_L)$. Since $B_L \subseteq P_L$, $|B_L|D \leqslant \varphi(\phi, P_L)$.

Since by the construction of algorithm 5, $I(B_1) = \frac{I(S_L)}{n^2}$, $|B_L|I(B_1) \leqslant n\frac{I(S_L)}{n^2} = \frac{I(S_L)}{n} \leqslant I(S_L) - \frac{I(S_L)}{n^2} = I(S_L) - I(B_1)$.

Let p_l be the leftmost point on B_1 and $l_l \in L$ the line corresponding to that point. Let p_1 be the leftmost point in S_L and let l_1 be the line corresponding to p_1.

Since both p_1 and p_l are on P_L, then $I(S_L) - I(B_1) \leqslant I(p_l, p_1) \leqslant I(p_l, \phi) + I(\phi, p_1) \leqslant \frac{\pi}{\sin\frac{\pi}{4}}(Dist(\phi, l_l) + Dist(\phi, l_1)) \leqslant \frac{\pi}{4\sin\frac{\pi}{4}}\varphi(\phi, P_L)$.

Thus, $\varphi(\phi, B) \leqslant \frac{2\epsilon}{\delta}|B_L|D + \frac{2\epsilon}{\delta}|B_L|I(B_1) \leqslant \frac{2\epsilon}{\delta}\varphi(\phi, P_L) + \frac{2\epsilon}{\delta}\frac{\pi}{4\sin\frac{\pi}{4}}\varphi(\phi, P_L) = \varphi(\phi, P_L)\frac{2\epsilon}{\delta}(1 + \frac{\pi}{4\sin\frac{\pi}{4}}) < 2.12\varphi(\phi, P_L)\frac{2\epsilon}{\delta}$

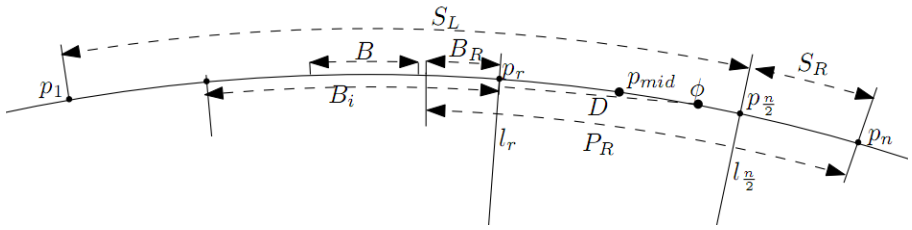

Figure 4.6: Proof of Lemma 4.2.8, case (ii).

(ii) If ϕ lies to the left of B, let B_R be the set of lines of B_1 that lies to the right of B. By observation 4.2.1 (ii), $|B| \leqslant \frac{2\epsilon}{\delta}|B_R|$. ϕ lies to the left of B therefore the distance from ϕ to each one of the points in B_R is greater than the distance from ϕ to each one of the points in B. Thus $\varphi(\phi, B) \leqslant \frac{2\epsilon}{\delta}\varphi(\phi, B_R) \leqslant \frac{2\epsilon}{\delta}\varphi(\phi, P_R)$.

Otherwise, if ϕ lies to the right of B or inside B, let $D = Dist(\phi, B_i)$ denote the distance between ϕ and the nearest line in B_i and let B_R denote the set of lines in B_i to the right of B. See Figure 4.6. As in previous case $\varphi(\phi, B) \leqslant |B|D + |B|I(B_i)$. As before by observation 4.2.1 (ii), we can bound the $|B|D$ by $\frac{2\epsilon}{\delta}\varphi(\phi, P_R)$. To bound $|B|I(B_i)$, let l_r and $l_{\lfloor n/2 \rfloor}$ be the right most lines of B_i and S_L, respectively, and let p_r and $p_{\lfloor n/2 \rfloor}$ be the intersection of those lines and C. Define p_{mid} to be the midpoint point on C between them.

By observations 4.2.1 (i) and (ii), $|B|I(B_i) \leqslant \frac{2\epsilon}{\delta}|B_R|2(I(p_r, p_{\lfloor n/2 \rfloor}))$. In case $p_{mid} \leqslant \phi$, we have $I(p_{\lfloor n/2 \rfloor}, p_r)/2 = I(p_{mid}, p_r) \leqslant I(\phi, p)$ for every point $p \in B_R$, and in case $p_{mid} > \phi$ we have $I(p_{\lfloor n/2 \rfloor}, p_r)/2 = I((p_{\lfloor n/2 \rfloor}, p_{mid}) \leqslant I(p, \phi)$ for every point $p \in S_R$. Since $B_R, S_R \subseteq P_R$, we conclude that, in any case, there are at least $|B_R|$ lines intersecting P_R that have a distance of at least $\frac{4}{\pi}\sin\frac{\pi}{4}I(p_{\lfloor n/2 \rfloor}, p_r)/2$ to ϕ. Thus, $|B_R|I(p_{\lfloor n/2 \rfloor}, p_r) \leqslant \frac{\pi}{2\sin\frac{\pi}{4}}\varphi(\phi, P_R)$, which gives us $|B||I(B_i)| \leqslant \frac{2\epsilon}{\delta}|B_R|2(I(p_r, p_{\lfloor n/2 \rfloor})) \leqslant \frac{8\epsilon}{\delta\sin\frac{\pi}{4}}\varphi(\phi, P_R)$. This suffices to conclude the proof of Lemma 4.2.8. □

To conclude the proof of Theorem 4.2.1 we summarize: we have shown that if $B \in B_1$,

$$|\varphi(\phi, B) - \varphi(\phi, S_B)| \leqslant (2 + \epsilon)\varphi(\phi, B) \leqslant 3\epsilon\varphi(\phi, P_L) = 3\epsilon\varphi(\Phi, P_L) \leqslant 3\epsilon\varphi(\Phi, L)$$

And for $B \notin B_1$,

$$|\varphi(\phi, B) - \varphi(\phi, S_B)| \leqslant (2 + \epsilon)\varphi(\phi, B) \leqslant 3\epsilon\varphi(\phi, P_R) = 3\epsilon\varphi(\Phi, P_R) \leqslant 3\epsilon\varphi(\Phi, L)$$

Algorithm 6 Coreset for lines.

1: Find Φ', $|\Phi'| \geqslant k$ such that $\varphi(\Phi', L) \leqslant 8\varphi^*(L)$.
2: Find Φ'', $|\Phi''| \geqslant |\Phi'|$ such that $\varphi(\Phi'', L) \leqslant 16\epsilon\varphi^*(L)$.
3: Move each line of L to its nearest facility in Φ'' resulting with L'. This will create a partition of lines in L into subsets, each subset L'_i of size n_i intersecting at a single point. We denote the geometrical structure of the subsets as "star shape".
4: For each star shaped subset L'_i find a strong coreset L''_i.
5: **return** $\cup L''_i$

It is left to analyze the running time of Algorithm 5:

Lemma 4.2.9. *The running time of Algorithm 5 is* $O(n\frac{c^k}{\epsilon^k}\log^{2k-1} n)$.

Proof. Consider the tree representing the recursive calls of Algorithm 5. The depth of the tree is exactly k (as in line 19 of Algorithm 5 the recursive call is with $k-1$). The number of leaves in the tree (i.e., executions with parameter $k-1$) is bounded by the number of points $|S|$ which is at most $\frac{c^k}{\epsilon^k}\log^{2k-2} n$. Each execution takes time $O(n \lg n)$ and each call to $k=1$ takes time $O(n \lg n)$ as seen in Claim 4.2.4. Thus, we conclude that the total running time is $O(n\frac{c^k}{\epsilon^k}\log^{2k-1} n)$. □

This concludes the proof of Theorem 4.2.1. □

Remark 4.2.1. *To be consistent with previous sections of the thesis, throughout we have used an approximation factor of* $\epsilon^{\frac{2}{3}}$. *We can change the approximation to be* $|\varphi(L') - \varphi(L)| \leqslant O(\epsilon'\varphi(L))$ *instead of* $|\varphi(L') - \varphi(L)| \leqslant O(\epsilon^{\frac{2}{3}}\varphi(L))$ *by picking* $\epsilon' = \epsilon^{\frac{2}{3}}$. *This will result in a strong coreset* L' *for* L *such that* $|\varphi(L') - \varphi(L)| \leqslant O(\epsilon'\varphi(L))$ *of size* $O(\frac{c^k}{\epsilon'^{\frac{3k}{2}}}\log^{2k-2} n)$ *for some constant* c. *The running time is* $O(n\frac{c^k}{\epsilon'^{\frac{3k}{2}}}\log^{2k-1} n)$

4.2.4 Strong corsets for the k-median problem for lines in $\mathbb{R}^2$

In the previous section we designed an algorithm that finds a strong coreset for CK-median (namely, for a set of lines that all intersected at a single point). We now extend our results for a general set of lines in the plane.

Theorem 4.2.2. *Let* $L = \{l_1, \ldots, l_n\}$ *be a set of lines in* $\mathbb{R}^2$. *We can find a strong coreset* L'' *of size* $O(\frac{c^{2k-1}k^2}{\epsilon^{3k}}\lg n \ln(\frac{k\lg n}{\epsilon})\log_{1+\epsilon}\frac{n}{\epsilon}\log^{4k-4} n)$ *for some constant* c *such that for each set of facilities* Φ, $|\varphi(\Phi, L) - \varphi(\Phi, L'')| \leqslant O(\epsilon\varphi(\Phi, L))$ *in running time* $O(n(\frac{c}{\epsilon^{\frac{3}{2}}})^{2k-1}\log^{4k-2} n)$.

Proof. We use the four step Algorithm 6. We will start by showing that the size of the coreset is $O(\frac{c^{2k-1}k^2}{\epsilon^{3k}}\lg n \ln(\frac{k\lg n}{\epsilon})\log_{1+\epsilon}\frac{n}{\epsilon}\log^{4k-4} n)$. The first three steps of the algorithm move each line (to finally form a number of *star shapes*, i.e., a set of lines intersecting at a single point) and do not change the number of lines. Since step 4 works on each star shape, the number of star shapes returned from step 3 defines the number of coresets return in step 4. By Lemma 4.1.5 the number of coresets is $O(\frac{k^2}{\epsilon}\lg n \ln(\frac{k\lg n}{\epsilon})\log_{1+\epsilon}\frac{n}{\epsilon})$. By Lemma 4.2.5 the number of lines of each coreset of size n_i is $O(\frac{c^{k'}}{\epsilon^{\frac{3k'}{2}}}\log^{2k'-2} n_i)$ for k' voronoi cells. Since we have $2k-1$ Voronoi cells, the number of lines of each coreset is of size n_i is $O(\frac{c^{2k-1}}{\epsilon^{\frac{6k-3}{2}}}\log^{4k-4} n_i)$. The union of those coresets is a coreset of size $O(\frac{c^{2k-1}k^2}{\epsilon^{3k}}\lg n \ln(\frac{k\lg n}{\epsilon})\log_{1+\epsilon}\frac{n}{\epsilon}\log^{4k-4} n)$ for some constant c.

Now we proof that the algorithm running time is $O(n(\frac{c}{\epsilon})^{2k-1}\log^{4k-2}n)$. Step 1 of the algorithm is executed in running time of $O(nk^2\ln(\frac{k\lg n}{\epsilon})\lg n)$ by Lemma 4.1.4. Steps 2 and 3 can be done in running time proportional to the number of returned points which is by Lemma 4.1.5 $O(n\frac{k^2}{\epsilon}\lg n\ln(\frac{k\lg n}{\epsilon})\log_{1+\epsilon}\frac{n}{\epsilon})$. Step 4's running time for $k'=2k-1$ Voronoi cells, is by Lemma 4.2.9 $O(n\frac{c^{k'}}{\epsilon^{\frac{3k'}{2}}}\log^{2k'-1}n)=O(n(\frac{c}{\epsilon^{\frac{3}{2}}})^{2k-1}\log^{4k-2}n)$. Thus the running time of the entire algorithm is dominated by $O(n(\frac{c}{\epsilon^{\frac{3}{2}}})^{2k-1}\log^{4k-2}n)$.

We are now left with showing that for each set of facilities Φ of size at most k, $|\varphi(\Phi,L'')-\varphi(\Phi,L)|\leqslant O(\epsilon\varphi(\Phi,L))$. In Steps 1,2 and 3 of Algorithm 6 we move the lines of L to create L'. By Lemma 4.1.7 $|\varphi(\Phi,L)-\varphi(\Phi,L')|\leqslant 32\epsilon\varphi^*(L)$. By Theorem 4.2.1 for each "star shape" L_i of size n_i we find a strong coreset L_i'' for the k-median problem, such that $|\varphi(\Phi,L_i'')-\varphi(\Phi,L_i')|\leqslant\epsilon\varphi(\Phi,L_i')$.

We now have, for any Φ, $|\varphi(\Phi,L)-\varphi(\Phi,L')|\leqslant 32\epsilon\varphi^*(L)\leqslant 32\epsilon\varphi(\Phi,L)$, thus $\varphi(\Phi,L')\leqslant(1+32\epsilon)\varphi(\Phi,L)$.

We conclude that

$$\begin{aligned}|\varphi(\Phi,L'')-\varphi(\Phi,L)| &\leqslant |\varphi(\Phi,L'')-\varphi(\Phi,L')|+|\varphi(\Phi,L)-\varphi(\Phi,L')|\\ &\leqslant \epsilon\sum_i\varphi(\Phi,L_i')+32\epsilon\varphi^*(L)\\ &\leqslant \epsilon(1+32\epsilon)\varphi(\Phi,L)+32\epsilon\varphi(\Phi,L)\\ &\leqslant 65\epsilon\varphi(\Phi,L)\end{aligned}$$

□

4.2.5 $(1+\epsilon)$ approximate clustering for k-median for lines in the plane

In this section, we use our coresets to solve k-median for lines in an approximate manner.

Theorem 4.2.3. *Let $L=\{l_1,\ldots,l_n\}$ be a set of lines in $\mathbb{R}^2$ and let $\epsilon<1$ then one can find a $(1+\epsilon)$- approximation to the k-median for lines in time $n(c/\epsilon)^{poly(k)}(\log n)^{poly(k)}$.*

Proof. We will use Algorithm 7

Algorithm 7 Finding a $1+\epsilon$ approximation to L.

1: $L''\leftarrow$ the lines returned from Algorithm 4
2: $P\leftarrow$ the points returned from Algorithm 2 when applied to L''.
3: **return** P

We claim that $P=\varphi^*(L'')$ is a $(1+\epsilon)$ approximation to the optimal solution for L, $\varphi^*(L)$. Let Φ_1 be the optimal solution for L and let Φ_2 be the optimal solution for L''. By Theorem 4.2.2 for each set of facilities Φ, $|\varphi(\Phi,L)-\varphi(\Phi,L'')|\leqslant O(\epsilon\varphi(\Phi,L))$. Then, $|\varphi(\Phi_1,L)-\varphi(\Phi_1,L'')|\leqslant O(\epsilon\varphi(\Phi_1,L))$ and $|\varphi(\Phi_2,L)-\varphi(\Phi_2,L'')|\leqslant O(\epsilon\varphi(\Phi_2,L))$. We also know that $\varphi(\Phi_1,L)\leqslant\varphi(\Phi_2,L)$ and $\varphi(\Phi_2,L'')\leqslant\varphi(\Phi_1,L'')$.

If $\varphi^*(L)\leqslant\varphi^*(L'')$ then

$$\begin{aligned}|\varphi^*(L)-\varphi^*(L'')| &= \varphi(\Phi_2,L'')-\varphi(\Phi_1,L)\\ &\leqslant \varphi(\Phi_1,L'')-\varphi(\Phi_1,L)\\ &= |\varphi(\Phi_1,L)-\varphi(\Phi_1,L'')|\\ &\leqslant O(\epsilon\varphi(\Phi_1,L)).\end{aligned}$$

If $\varphi^*(L) > \varphi^*(L'')$ then for any constant c, $(1 - c \cdot \epsilon)\varphi(\Phi_1, L) \leqslant (1 - c \cdot \epsilon)\varphi(\Phi_2, L)$. Thus, $c \cdot \epsilon\varphi(\Phi_2, L) \leqslant \varphi(\Phi_2, L) - \varphi(\Phi_1, L) + c \cdot \epsilon\varphi(\Phi_1, L)$. Now, let c be a constant such that $|\varphi(\Phi_2, L) - \varphi(\Phi_2, L'')| < c \cdot \epsilon\varphi(\Phi_2, L)$:

$$\begin{aligned}
|\varphi^*(L) - \varphi^*(L'')| &= \varphi(\Phi_1, L) - \varphi(\Phi_2, L'') \\
&= \varphi(\Phi_1, L) - \varphi(\Phi_2, L) + \varphi(\Phi_2, L) - \varphi(\Phi_2, L'') \\
&\leqslant \varphi(\Phi_1, L) - \varphi(\Phi_2, L) + c \cdot \epsilon\varphi(\Phi_2, L) \\
&\leqslant \varphi(\Phi_1, L) - \varphi(\Phi_2, L) + \varphi(\Phi_2, L) - \varphi(\Phi_1, L) + c \cdot \epsilon\varphi(\Phi_1, L) \\
&= O(\epsilon\varphi(\Phi_1, L)).
\end{aligned}$$

The running time of line 1 of Algorithm 7 is by Lemma 4.2.2 $O(n(\frac{c}{\epsilon^{\frac{3}{2}}})^{2k-1}\log^{4k-2} n)$. The running time of line 2 is $O(\bar{n}^{2k}\log\bar{n})$ where $\bar{n}$ is the number of lines returned from Algorithm 4. The value of $\bar{n}$ by Lemma 4.2.5 is at most $(\frac{c}{\epsilon^{\frac{3}{2}}})^{2k-1}\log^{4k-4} n$ for some large enough constant c.

This gives a total running time of $n(c/\epsilon)^{poly(k)}(\log n)^{poly(k)}$. Here, $poly(k)$ is some polynomial in k.

□

Bibliography

[AGI+92] R. Agrawal, S. Ghosh, T. Imielinski, B. Iyer, and A. Swami. An interval classifier for database mining applications. In *VLDB '92: Proceedings of the 18th International Conference on Very Large Data Bases*, pages 560–573, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[All02] P. D. Allison. *Missing Data*. Sage Publications, 2002.

[Ber06] P. Berkhin. A survey of clustering data mining techniques. pages 25–71. Springer, 2006.

[BMM03] P. Bose, A. Maheshwari, and P. Morin. Fast approximations for sums of distances, clustering and the Fermat–Weber problem. *Comput. Geom. Theory Appl.*, 24(3):135–146, 2003.

[CEK02] J. F. Campbell, A. T. Ernst, and M. Krishnamoorthy. *Facility Location: Applications and Theory*, chapter Hub Location Problems, pages 373–407. Springer Verlag, 2002.

[Che06] K. Chen. On k-median clustering in high dimensions. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1177–1185, New York, NY, USA, 2006. ACM.

[DRSS96] A. A. Diwan, S. Rane, S. Seshadri, and S. Sudarshan. Clustering techniques for minimizing external path length. In *VLDB '96: Proceedings of the 22th International Conference on Very Large Data Bases*, pages 342–353, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[FFS06] D. Feldman, A. Fiat, and M. Sharir. Coresets for weighted facilities and their applications. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 315–324, Washington, DC, USA, 2006. IEEE Computer Society.

[FFSS07] D. Feldman, A. Fiat, D. Segev, and M. Sharir. Bi-criteria Linear-time Approximations for Generalized k-Mean/Median/Center. In *Proc. 23th Annu. ACM Symposium on Computational Geometry (SoCG)*, pages 19–26, 2007.

[FG88] T. Feder and D. Greene. Optimal algorithms for approximate clustering. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 434–444, New York, NY, USA, 1988. ACM.

[FMS07] D. Feldman, M. Monemizadeh, and C. Sohler. A ptas for k-means clustering based on weak coresets. In *SCG '07: Proceedings of the twenty-third annual symposium on Computational geometry*, pages 11–18, New York, NY, USA, 2007. ACM.

[GLS06] J. Gao, M. Langberg, and L. Schulman. Analysis of incomplete data and an intrinsic-dimension helly theorem. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 464–473, New York, NY, USA, 2006. ACM.

[GLS10] J. Gao, M. Langberg, and L. J. Schulman. Clustering lines: classification of incomplete data. *ACM Transactions on Algorithms*, 7:8, 2010.

[HP06] S. Har-Peled. How to get close to the median shape. In *SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry*, pages 402–410, New York, NY, USA, 2006. ACM.

[HPK05] S. Har-Peled and A. Kushal. Smaller coresets for k-median and k-means clustering. In *SCG '05: Proceedings of the twenty-first annual symposium on Computational geometry*, pages 126–134, New York, NY, USA, 2005. ACM.

[HPM04] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300, New York, NY, USA, 2004. ACM.

[MS84] N. Megiddo and K. J. Supowit. On the complexity of some common geometric location problems. *SIAM J. Comput.*, 13:182–196, 1984.

[MT82] N. Megiddo and A. Tamir. On the complexity of locating linear facilities in the plane. *Operations Research Letters*, 13:194–197, 1982.

[Pap81] C. H. Papadimitriou. Worst-case and probabilistic analysis of a geometric location problem. *SIAM Journal of Computing*, 10:542 – 557, 1981.

[SA96] M. Sharir and P.K. Agarwal. *Davenport-Schinzel sequences and their geometric applications*. Cambridge University Press, New York, NY, USA, 1996.

[Woe03] G. J. Woeginger. Exact algorithms for NP-hard problems: a survey. *Lecture Notes in Computer Science*, 2570/2003:185–207, 2003.

[WWP00] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *ECCV*, LNCS 1842, pages 18–32. Springer Verlag, 2000.

Appendices

Appendix A

Finding line k-median for points on a line

In this section we present an algorithm for (exactly) finding the optimal k-median solution for points on a line. Our algorithm is used in Section 2.2, after the proof of Lemma 2.2.1. Our alorithm is based on dynamic programming, and is first presented in an abstract manner, afterwhich we elaborate on the application to points on a line.

A.1 General Algorithm

Let U be a set of input elements. Let V be a set of potential "center" elements. Let d be a distance function $d : U \times V \to \mathbb{R}^+$.

For sets $A = \{a_1, \ldots, a_n\} \subset U$ and $M = \{m_1, \ldots, m_k\} \subset V$ let $d(a_i, M) = \min_{j=1\ldots k} d(a, m_j)$. Let $d(A, M) = \sum_{i=1}^{n} d(a_i, M)$. Let $B = \{B_1, \ldots, B_k\}$ be a partition of A such that $B_j = \{a_i | d(a_i, M) = d(a_i, m_j)\}$. Finally, let $\varphi(A, k) = \min_{\|M\|=k} d(A, M)$.

Theorem A.1.1. *If we can order A in such a way that*

1. *For any set M, in the corresponding partition $B = \{B_1, ..., B_k\}$, each set B_j takes the form $\{a_r, a_{r+1}, \ldots, a_s\}$ for some $r \leqslant s$.*
2. *$\cup_j B_j = A$.*
3. *There exists an algorithm that finds $\varphi(\{a_r, a_{r+1}, \ldots, a_s\}, 1)$ for any $r \leqslant s$.*

Then, we can find $\varphi(A, k)$ in time $O(n^2kh)$, where h is the running time of the algorithms for finding $\varphi(\{a_r, \ldots, a_s\}, 1)$ in (3) above.

Proof. We will present an algorithm that computes $\varphi(A, k)$ using dynamic programming. Let σ be a two dimensional $n \times k$ array, each entry $\sigma(i, j)$ of the array, describes the cost of $\varphi(\{a_1, \ldots, a_i\}, j)$. Our goal is to compute $\varphi(A, k) = \sigma(n, k)$.

Each cell of the array is computed as follows:

$$\sigma(i, j) = \min_{r=1,\ldots,i-1} (\varphi(\{a_{r+1}, \ldots, a_i\}, 1) + \sigma(r, j - 1)) \tag{A.1}$$

We now show by induction that the above definition of $\sigma(i, j)$ indeed equals $\varphi(\{a_1, \ldots, a_i\}, j)$.

Lemma A.1.1. $\varphi(\{a_1, \ldots, a_i\}, j) = \min_{r=1,\ldots,i-1}(\varphi(\{a_{r+1}, \ldots, a_i\}, 1) + \varphi(\{a_1, \ldots, a_r\}, j - 1)$.

Proof. Let M be a set realizing $\varphi(\{a_1,\ldots,a_i\},j)$. Namely, $M=\{m_1,\ldots,m_j\}$. Let B_j be the partition defined by M. It holds by our assumptions that $B_j=\{a_{r+1},\ldots,a_i\}$ for some $r\leqslant i-1$. Thus,

$$\varphi(\{a_1,\ldots,a_i\},j)=\sum_{\ell=1}^{r} d(a_\ell,M\backslash\{m_j\})+\varphi(\{a_{r+1},\ldots,a_i\},1)\geqslant\varphi(\{a_1,\ldots,a_r\},j-1)+\varphi(\{a_{r+1},\ldots,a_i\},1).$$

On the other hand, as $\varphi(\{a_1,\ldots,a_i\},j)=\min_{\|M\|=k} d(A,M)$ it holds that $\varphi(\{a_1,\ldots,a_i\},j)\leqslant\varphi(\{a_1,\ldots,a_r\},j-1)+\varphi(\{a_{r+1},\ldots,a_i\},1)$. Therefore $\exists r\leqslant i-1$ such that $\varphi(\{a_1,\ldots,a_i\},j)=\varphi(\{a_1,\ldots,a_r\},j-1)+\varphi(\{a_{r+1},\ldots,a_i\},1)$. Since r minimizes the value of the right side of the equation $\varphi(\{a_1,\ldots,a_i\},j)=\min_{r=1,\ldots,i-1}(\varphi(\{a_{r+1},\ldots,a_i\},1)+\varphi(\{a_1,\ldots,a_r\},j-1)$

□

The only thing left now to show is how to initialize the array. In order for our algorithm to work we need to initialize all $\sigma(i,1)$ for $1\leqslant i\leqslant k$. Since one of the requirements for A is that we can compute each $\varphi(\{a_j,\ldots,a_{j+i}\},1)$, we can also compute each $\varphi(\{a_1,\ldots,a_i\},1)$ which is the value of $\sigma(i,1)$.

□

If we also wants to find the partition B corresponding to the optimal solution , it can be done by keeping an additional two dimensional $n\times k$ array, $\psi(i,j)$, where each entry describes B for $\{a_1,\ldots,a_i\}$ and a set of optimal centers M of size j, that are used to create $\varphi(\{a_1,\ldots,a_i\},j)=\min_{\|M\|=j} d(\{a_1,\ldots,a_i\},M)$.

A.2 Algorithm for points on a line

Lemma A.2.1. *Let $P=\{p_1,\ldots,p_n\}$ be a set of n points on a line l. Let $\Phi=\{\phi_1,\ldots,\phi_k\}$ be a set of k points. For a point $p\in P$ let $\varphi(\Phi,p)=\min_{\phi\in\Phi} Dist(p,\phi)$ and let $\varphi(\Phi,P)=\sum_{p\in P}\varphi(\Phi,p)$. Let $\varphi(P,k)=\min_{\|\Phi\|=k}\varphi(\Phi,P)$ be the minimum solution, then we can find $\varphi(P,k)$ in $O(n^3k)$.*

Proof. We can apply the previous algorithm in order to find $\varphi(P,k)$ by setting $A=P$ and $M=\Phi$. By Lemma 2.1.1 the optimal solution can contain only points from P, therefore we can set $V=P$. We now prove that the set P can be ordered according to the constrains of Theorem A.1.1. Since all the points in P are on the same line, we can order them on the line such that p_1 is the left most point and p_n the right most point.

For each set of facilities Φ we can create partitions such that each partition contains points $\{p_r,p_{r+1},\ldots,p_s\}\subset P$ for some $r\leqslant s$. For a partition $\{p_r,p_{r+1},\ldots,p_s\}$, the optimal solution is $\varphi(\{p_r,\ldots,p_s\},1)=\sum_{p\in\{p_r,\ldots,p_s\}} Dist(p,p_{\lfloor\frac{s-r}{2}\rfloor})$, which can be found in time $O(n)$. By Theorem A.1.1, we can find $\varphi(P,k)$ in time $O(n^3k)$. □

Appendix B

Properties of the circle

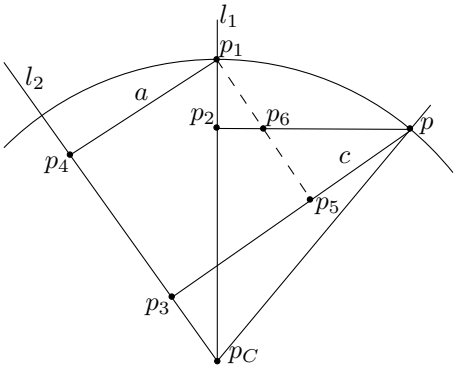

Figure B.1: Proof of Lemma B.0.2.

Lemma B.0.2. *Let C be a circle with center p_C, let l_1 and l_2 be two lines passing trough p_C and let p be another point on C. Let p_1 be the intersection point of l_1 and C that is closer to p. Then $Dist(p, l_1) + Dist(p_1, l_2) \geqslant Dist(p, l_2)$.*

Proof. We start with some notations. Let p_2 be the closest point on l_1 to p, let p_3 be the closest point on l_2 to p and let p_4 be the closest point on l_2 to p_1. See Figure B.2.

If $Dist(p, l_1) \geqslant Dist(p, l_2)$ then we are done. Otherwise, let $a = Dist(p_1, p_4)$ and $b = Disp(p, p_3)$. Since both (p_1, p_4) and (p, p_3) are perpendicular to l_2 they are parallel to each other. Thus we can pass a line from p_1 parallel to l_2 intersecting (p, p_3) at point p_5 and (p_2, p) at point p_6 (If it does not intersects (p, p_3) or (p_2, p) then $Dist(p_1, l_2) \geqslant Dist(p, l_2)$). $Dist(p_1, p_4) = Dist(p_5, p_3)$. If we look at the triangle p, p_5, p_6, $Dist(p, p_6) > Dist(p, p_5)$ thus $Dist(p, l_1) + Dist(p_1, l_2) > Dist(p, p_6) + Dist(p_1, p_4) > Dist(p, p_5) + Dist(p_1, p_4) = Dist(p, p_5) + Dist(p_5, p_3) = Dist(p, l_2)$. □

Lemma B.0.3. *Let C be a circle with radius* 1 *and center p_C, let $L = \{l_1, \ldots, l_n\}$ be a set of lines passing trough p_C and let p_1 and p_2 be another two points on C. Let $\bar{l}_1$ be a line passing trough p_1 and p_C and $\bar{l}_2$ be a line passing trough p_2 and p_C. Let β be the smallest angle between $\bar{l}_1$ and $\bar{l}_2$, assume that p_2 is to the* right *of p_1 in clockwise order via β. For a line $l_i \in L$ let α_i be the angle between l_i and $\bar{l}_1$ and γ_i be the angle between l_i and $\bar{l}_2$. Let L_1 be the set of lines that lie between $\bar{l}_1$ and $\bar{l}_2$, let $L_2 \subset L$ be the set of lines that lies between a line perpendicular to $\bar{l}_1$ and a line perpendicular to $\bar{l}_2$, let $L_3 \subset L$ be all the lines between $\bar{l}_1$ and the line perpendicular to $\bar{l}_2$ and let $L_4 \subset L$ be all*

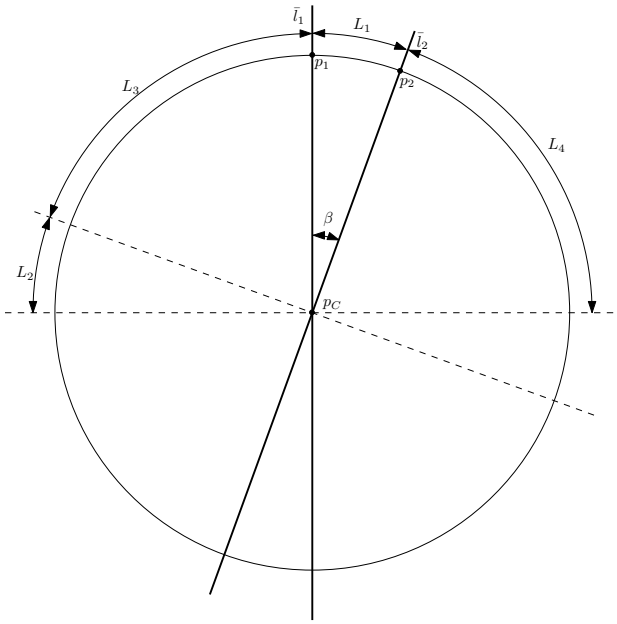

Figure B.2: Proof of Lemma B.0.3.

the lines between $\bar{l}_2$ and the line perpendicular to $\bar{l}_1$. Note that $L = L_1 \cup L_2 \cup L_3 \cup L_4$. See Figure B.2. *For a point p, let l_p be the line passing through p and p_C. Let $\varphi(L,p) = \sum_{l \in L} Dist(l,p)$ and let $Z(p) = \sum_{l_i \in L} \cos\alpha_i(p)$. Here $\alpha_i(p)$ is the angle between l_p and l_i. Let $L(p)$ be the set of lines that lie to the* left *of p in counterclockwise direction between the line l_p and the line perpendicular to l_p. It holds that $L(p_1) = L_2 \cup L_3$, and $L(p_2) = L_1 \cup L_3$. Let $\varphi_{L(p)}(p) = \sum_{l \in L(p)} Dist(l,p)$ and let $Z_{L(p)}(p) = \sum_{l_i \in L(p)} \cos\alpha_i(p)$.*

If we know $\varphi(L,p_1)$, $\varphi_{L(p_1)}(p_1)$, $Z(p_1)$, $Z_{L(p_1)}(p_1)$ and the set of lines L_1 and L_2 then we can compute $\varphi(L,p_2)$, $\varphi_{L(p_2)}(p_2)$, $Z(p_2)$ and $Z_{L(p_2)}(p_2)$ in time $O(|L_1| + |L_2|)$.

Proof. Since the radius of C is 1 it holds that $\varphi(L,p_1) = \sum_{l_i \in L} Dist(l_i,p_1) = \sum_{l_i \in L} \sin\alpha_i$. We will start by computing $\varphi_{L(p_2)}(p_2)$ and $Z_{L(p_2)}(p_2)$. By definition it holds that $\varphi_{L(p_2)}(p_2) = \sum_{l_i \in L_3} \sin\gamma_i + \sum_{l_i \in L_1} \sin\gamma_i$ and that $Z_{L(p_2)}(p_2) = \sum_{l_i \in L_3} \cos\gamma_i + \sum_{l_i \in L_1} \cos\gamma_i$.

The sums $\sum_{l_i \in L_1} \sin\gamma_i$ and $\sum_{l_i \in L_1} \cos\gamma_i$ can be calculated in time $O(|L_1|)$. We will now present a way to calculate $\sum_{l_i \in L_3} \sin\gamma_i$ and $\sum_{l_i \in L_3} \cos\gamma_i$ in time $O(|L_2|)$

$$\begin{aligned}
\sum_{l_i \in L_3} \sin\gamma_i &= \sum_{l_i \in L_3} \sin(\alpha_i + \beta) \\
&= \sum_{l_i \in L_3} \sin\alpha_i \cos\beta + \sum_{l_i \in L_3} \cos\alpha_i \sin\beta \\
&= \cos\beta \sum_{l_i \in L_3} \sin\alpha_i + \sin\beta \sum_{l_i \in L_3} \cos\alpha_i
\end{aligned}$$

$$\begin{aligned}\sum_{l_i\in L_3}\cos\gamma_i &= \sum_{l_i\in L_3}\cos(\alpha_i+\beta)\\ &= \sum_{l_i\in L_3}\cos\alpha_i\cos\beta-\sum_{l_i\in L_3}\sin\alpha_i\sin\beta\\ &= \cos\beta\sum_{l_i\in L_3}\cos\alpha_i+\sin\beta\sum_{l_i\in L_3}\sin\alpha_i\end{aligned}$$

We now notice that $\sum_{l_i\in L_3}\sin\alpha_i=\varphi_{L(p_1)}(p_1)-\sum_{l_i\in L_2}\sin\alpha_i$ and $\sum_{l_i\in L_3}\cos\alpha_i=Z_{L(p_1)}(p_1)-\sum_{l_i\in L_2}\cos\alpha_i$, thus we can compute both expression in time of $O(|L_2|)$. Therefor the total running time of computing $\varphi_{L(p_2)}(p_2)$ and $Z_{L(p_2)}(p_2)$ is $O(|L_1|+|L_2|)$.

We will now present a way to compute $\varphi(p_2)$ and $Z(p_2)$ is $O(|L_1|+|L_2|)$.

- For the lines of $l_i\in L_1$, $\gamma_i=\beta-\alpha_i$.
- For the lines of $l_i\in L_2$, $\gamma_i=\pi-(\beta+\alpha_i)$.
- For the lines of $l_i\in L_3$, $\gamma_i=\alpha_i+\beta$.
- For the lines of $l_i\in L_4$, $\gamma_i=\alpha_i-\beta$.

Thus,

- For $l_i\in L_1$, $\sin\gamma_i=-\sin\alpha_i\cos\beta+\cos\alpha_i\sin\beta$
- For $l_i\in L_2$, $\sin\gamma_i=\sin\alpha_i\cos\beta+\cos\alpha_i\sin\beta$
- For $l_i\in L_3$, $\sin\gamma_i=\sin\alpha_i\cos\beta+\cos\alpha_i\sin\beta$
- For $l_i\in L_4$, $\sin\gamma_i=\sin\alpha_i\cos\beta-\cos\alpha_i\sin\beta$

Thus,

$$\begin{aligned}\varphi(L,p_2) &= \cos\beta\cdot\sum_{l_i\in L}\sin\alpha_i-2\cos\beta\cdot\sum_{l_i\in L_1}\sin\alpha_i+\sin\beta\cdot\sum_{l_i\in L}\cos\alpha_i-2\sin\beta\cdot\sum_{l_i\in L_4}\cos\alpha_i\\ &= \cos\beta\cdot\varphi(L,p_1)-2\cos\beta\cdot\sum_{l_i\in L_1}\sin\alpha_i+\sin\beta\cdot Z(p_1)-2\sin\beta\cdot\sum_{l_i\in L_4}\cos\alpha_i\end{aligned}$$

However, $\sum_{l_i\in L_4}\cos\alpha_i=Z(p_1)-Z_{L(p_1)}(p_1)-\sum_{l_i\in L_1}\cos\alpha_i$.

- For $l_i\in L_1$, $\cos\gamma_i=\cos\alpha_i\cos\beta+\sin\alpha_i\sin\beta$
- For $l_i\in L_2$, $\cos\gamma_i=-\cos\alpha_i\cos\beta+\sin\alpha_i\sin\beta$
- For $l_i\in L_3$, $\cos\gamma_i=\cos\alpha_i\cos\beta-\sin\alpha_i\sin\beta$
- For $l_i\in L_4$, $\cos\gamma_i=\cos\alpha_i\cos\beta+\sin\alpha_i\sin\beta$

$$\begin{aligned}Z(p_2) &= \cos\beta\cdot\sum_{l_i\in L}\cos\alpha_i-2\cos\beta\cdot\sum_{l_i\in L_2}\cos\alpha_i+\sin\beta\cdot\sum_{l_i\in L}\sin\alpha_i-2\sin\beta\cdot\sum_{l_i\in L_3}\sin\alpha_i\\ &= \cos\beta\cdot Z(p_1)-2\cos\beta\cdot\sum_{l_i\in L_2}\cos\alpha_i+\sin\beta\cdot\varphi(L,p_1)-2\sin\beta\cdot\sum_{l_i\in L_3}\sin\alpha_i\end{aligned}$$

As before, $\sum_{l_i \in L_3} \sin \alpha_i = \varphi_{L(p_1)}(p_1) - \sum_{l_i \in L_2} \sin \alpha_i$. Thus to complete the computations of $Z(p_2)$ and $\varphi(L, p_2)$ the only expressions we need to compute are $\sum_{l_i \in L_1} \cos \alpha_i$, $\sum_{l_i \in L_1} \sin \alpha_i$, $\sum_{l_i \in L_2} \cos \alpha_i$ and $\sum_{l_i \in L_2} \sin \alpha_i$ which require running time of $O(|L_1| + |L_2|)$.

□

List of Figures

תוכן העניינים

תקציר

יהי $P = \{p_1,...,p_n\}$ קבוצה של נקודות ב R^d. בעיית ה k-median היא מציאת k נקודות (מרכזים) כך שסכום המרחקים מכל נקודה ב P למרכז הקרוב ביותר יהיה מינימלי. בעיית ה k-median הינה בעיה NP קשה. אלגוריתמי קירוב לבעיית ה k-median נחקרו רבות בעשור האחרון, וכיום ידוע אלגוריתם הרץ בזמן לינארי אשר מחזיר קבוצת המרכזים המקרבת את הפתרון האופטימלי בפקטור של $(1+\epsilon)$ עבור ϵ קבוע.

בעבודה זו אנו לומדים את בעיית ה k-median על קווים. יהי $L = \{l_1,...,l_n\}$ קבוצה של קווים. בעיית ה k-median עבור קווים, היא מציאת k נקודות (מרכזים) כך שסכום המרחקים מכל קו ב L למרכז הקרוב ביותר יהיה מינימלי.

בעבודה זו אנו מציגים הן אלגוריתם מדויק והן אלגוריתם קירוב לבעיית ה k-median עבור קווים ומתרכזים בעיקר במקרה בו הקווים הינם ב-R^2. אנו מראים כי ניתן לפתור את בעיית ה k-median עבור קווים בצורה יעילה עבור ערכים קבועים של k, אך היא NP-שלמה אם k הוא לא קבוע. יתר על כן, אנו מראים כי פתרון בעיית ה k-median עבור קווים ב-R^2 הרץ בזמן מהיר מ-$poly(n)2^{\epsilon k}$ משמעותו שניתן לפתור את בעיית SAT בזמן ריצה של $2^{\epsilon n}$, ולכן מציאת אלגוריתם הרץ בזמן $poly(n)2^{\epsilon k}$ לבעיית ה k-median עבור קווים כאשר ϵ הינו קטן, אינו סביר. לעומת זאת אנו מציגים אלגוריתם למציאת 1-median עבור קווים ב-R^2 בזמן ריצה $O(n^2 \lg n)$, ואלגוריתם למציאת k-median עבור קווים ב-R^2 בזמן ריצה $O(n^{2k} \lg n)$.

לאחר מכן, אנו פונים להציג אלגוריים קירוב לבעיית ה k-median עבור קווים. באמצעות הרעיונות של קירוב bi-criteria ומושג ה-coreset, אנו מתכננים אלגוריתם קירוב בזמן $n(c/\epsilon)^{poly(k)}\log(n)^{poly(k)}$ אשר מחזיר קירוב לבעיית ה k-median עבור קווים במישור (כאן c הינו קבוע מספיק גדול).

האוניברסיטה הפתוחה
המחלקה למתמטיקה ולמדעי המחשב

הצברה של קווים

עבודת תזה זו הוגשה כחלק מהדרישות לקבלת תואר
"מוסמך למדעים" M.Sc. במדעי המחשב
באוניברסיטה הפתוחה
החטיבה למדעי המחשב

על-ידי
תומר פרץ

העבודה הוכנה בהדרכתו של ד"ר מיכאל לנגברג

ינואר 2011